IQLR

International QL Report

ISSN 1078-5787

Volume 5 Issue 6
March/April
1996

The Natural Upgrade path for QDOS

SMSQ/E at WORK

On (4) Different Computers

(See page 56 for details)

E-Mail: iqlr@nccnet.com

IQLR.....

IQLR
P.O. Box 3991
Newport, RI 02840-0987
USA

IQLR
23 Ben Oakley Drive
Thetford, Norfolk IP24 1QJ
GREAT BRITAIN

Tel/Fax: +1 401 849 3805 E-Mail: iqlr@ncconet.com

PUBLISHER: Robert Dyl, Sr. (ISSN 1078-5787)

IQLR is published bi-monthly, our volume year begins on 1 May and runs through 30 April. Subscriptions begin with the current issue at the time of sign up. Subscription rates are as follows:

USA	$30.00 per year
British Isles & N. Ireland	£27.00 per year
Europe	£30.00 per year
Canada	$34.00 (US Funds)
Central/South America	$38.00 (US Funds)
Rest of World	$45.00 (US Funds)

UK and European readers may send their subscriptions to our European office listed above. Postal, Euro, Bank and Personal Cheques in Pounds Sterling, drawn on a UK bank should be made payable to IQLR.

Payment in US $ can be made by either a Postal, Bank or Personal cheque (drawn on a US BANK) or bank notes (£ or DM equivalent to the US $ amount) should be sent to our North American office.

Credit Card Holders may subscribe by either calling or sending their Credit Card number and Expiry date to: DI-REN - 59 William Street - Walsall WS4 2AX - UK Telephone/Fax: (44) (0) 1922 33580

Please note: DO NOT send Credit Card subscriptions to our North American office. Credit Cards will be charged in (£) Pounds Sterling.

We welcome your comments, suggestions and articles. YOU make IQLR possible. We are constantly changing and adjusting to meet your needs and requirements. Articles submitted for publication should be on a 3.5" disk in Quill or Text87 format. To enhance your article you may wish to include Saved Screen dumps. PLEASE send a hard copy of all screens to be included, don't forget to specify where in the text you would like the screens placed.

Article and Advertising DEADLINES are as follows:

Issue 1	10 April
Issue 2	10 June
Issue 3	10 August
Issue 4	10 October
Issue 5	10 December
Issue 6	10 February

IQLR is produced using the following equipment: either of 2 Tower Cased QL's with Super Gold Card, Qubide Hard Disk Interface, superHermes and/or Di-Ren Keyboard Interface and Minerva MK II. Masters are produced using a Hewlett Packard DeskJet 520 and an Epson 800+ InkJet printers.

Contents.....

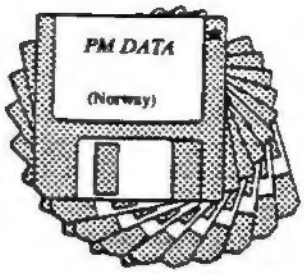
Visit IQLR's Web Page at:

http://www.forthrt.com/~di-ren/iqlr.html

Advertisers.....

DISK MATE 5 - Version 5.07

Massapequa, New York, USA - Bob Gilder

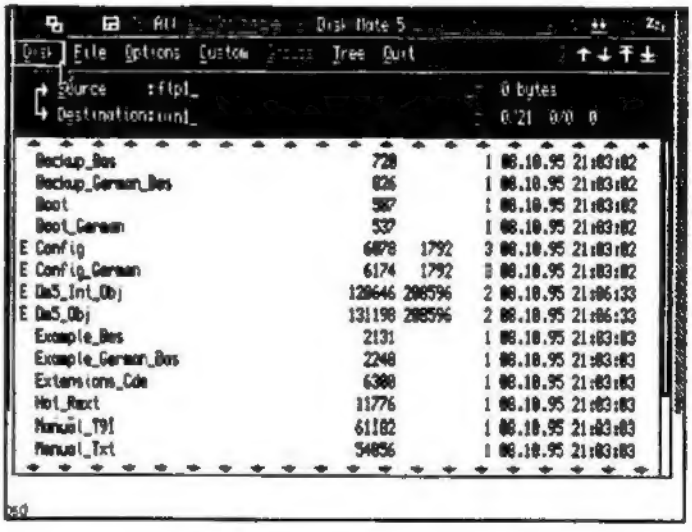


At the request of Bob Dyl, I have found myself reviewing another disk utility. However, I really didn't know what I was getting into. You see, Disk Mate 5 operates under the Pointer Environment and I have had only limited experience with the Pointer Environment some six or seven years ago with QRAM! Last year I purchased the QUANTA mouse interface and mouse from John Taylor, the Quanta Treasurer. At last, I will finally find out if it really works.

The Disk Mate 5 package contains the program disk and a comprehensive manual with many examples and I thank the author, Pal Monstad for providing these examples which helped me get to know the Pointer Environment and Disk Mate 5. From now on Disk Mate 5 will be written as DM5 and the Pointer Environment as PE.

The program disk contains quite a few program extensions such as those that compose the PE and there is a backup_bas program for making duplicates of DM5, Config which allows the user to make permanent changes to the default drive designators from either win_1 or flp1_ to RAM, FLP and MDV (1 through 8), and a Basic program Examples_bas and an Updates_text file. The main program files; DM5_obj, DM5_int_obj and Extensions_Cde. There are duplicates in German for most of these files.

The first item on hand was to make two backup copies of DM5 and placed the original in a safe place. I booted up one copy of DM5 and with a minute if came face to face with the DM5 opening screen. Scanning the screen, I noticed that there were two default drive designators, win1_ as the source and flp1_ as the designation. Playing around with some of the buttons, I found that I could change the drive defaults by pressing F3 or the button next to F3. Walking the mouse around the screen I happened to place the arrow on the vertical end of both drive designators and a vertical rectangle appeared. HiTting either one of the mouse buttons made the default designators change or swapping the source and designation of the drives. In the manual it states that there were provisions for changing drive designators. At this time, I didn't want to press my luck since things seem to be going my way!



If you move the pointer to one of the drive designators and then HIT or DO, another window pops up. This is used to make a change for the drive designator. The left-hand side of the window displays RAM FLP MDV and a HIT on any one of these drive designators will replace the original default designator. At the center of the window are numbers 1 through 8 which can be selected by a HIT on any number and it will be appended to the new drive designator and at this time, if you like, you can select this drive to be either the Data Default or Prog Default by moving the pointer on either of these selections with a HIT on the selected default! Move the pointer to the upper right to the OK button and if you are satisfied with your selections, HIT OK. This change of drive designator will hold through the entire session of DM5. This is only temporary. The Config program (EXEC the program to use) displays the identical window as the previous window in DM5 for changing the drive designators, however, this will be a permanent drive designator change.

At this time, the pointer should be placed on the DISK button which is situated at the left-hand section of the screen and then HIT it. A pull down menu appears with the following functions:

Directory, Format, Disk info, Sector copy disk and in light green print, Print directory. Since we have not selected Directory, we can not Print directory - so HIT the Directory line and the source directory will appear

below the Menu Line. If you HIT the Directory button again you will see that the Print directory function is now with black print and if you would like, the directory can be printed from your printer.

If you select Format, another pull down menu appears with the following:

ESC OK
Device : flp2_ DD HD ED
Disk Name :
Capacity : Minimum

The word Minimum is printed in light green - if you require a meaning for minimum, you could ESCape out of the Format/Disk function and place the pointer at the ALL button, HIT it and the directory will change color from black background with white letters to a green background with black letters. Then navigate to the Disk button; HIT it and then place the pointer at Format and all text within the format window is black. The default disk capacity is DD and this will be colored with green. Lower the pointer to Disk name and a box will appear; HIT it! A cursor will appear and you can write in a name up to ten characters for the disk name. DO it and the pointer will be on ESC; move the pointer to OK and HIT it and the disk will be formatted. When the format has completed, ESCape out of the Format function.

If you would like to see a wide range of disk information about your disk, you can again HIT the Disk button and move the pointer to Disk info and HIT it. The next option or function within the Disk menu is 'Sector copy disk'. This command is only available if both drives are floppy disks and the menu is as follows:

ESC OK
DD HD ED Error check

The default disk format is DD. When you are ready to make an exact duplicate diskette from the source disk, then place the pointer to OK and HIT. Another window appears stating 'Please wait' and has an icon of a steaming cup of coffee or tea - its' meaning that it takes a few minutes to copy a disk so .... get a cup of java while you are waiting for the copying completion. Below the icon is a green bar which will serve as the total sector of the disk. As the disk is copied, the green bar is overlaid with red as each sector on the target disk is copied.

Incidentally, I like the idea of using a sector copier for all my disk copies. One benefit of using a sector copier is that any disk which has been QL formatted and say, an old duplicate of a program which is not used any more, can be used with a sector copier - it becomes overwritten without formatting the disk all over again.

I have tried a test for the Sector copier by using a DD disk with bad tracks. When the Sector copier advanced to track 66 on the disk, the copying stopped, and another menu advised that there was a problem with the copying process, either try again or Abort! This is the way a copier should operate; it will only copy a disk with ALL GOOD tracks.

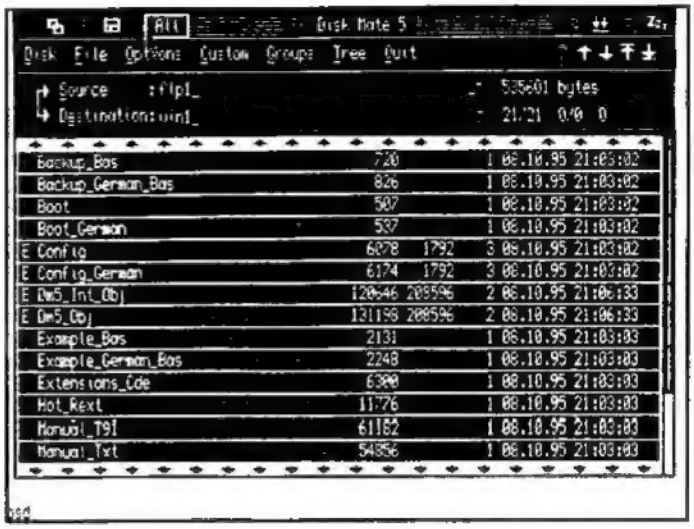
The Print directory is the last function on the Disk menu and as long as your printer is Epson compatible, you should get a hard copy of your disk directory. If you would like to send some additional control characters to your printer, you will have to place the pointer to the Custom menu and HIT it. Refer to section 4.4 for the necessary information on how it is accomplished. The file menu offers quite a few functions and they are as follows:

ESC
Copy files
Make sub-dir
Delete files
* Delete sub-dir (this is light green, will change when sub-dir made
Rename replace
Rename erase
Rename add
Convert

DISK MATE 5 - (cont'd)

The Copy files command copies files from the source drive to the destination device. The destination device can be RAM, MDV or FLP, so make sure that your destination device is actually the device you want your files to be copied on. If you want to have additional copies made , HIT the Multiple button on the copy menu. There is an Overwrite button within this menu if the files on the destination disk has identical files to be updated. After files have been copied to your destination drive there will be a request if you want to make more copies. Yes will continue the copy routine; No will exit from this menu.

Note: The same disk that I used to test the Sector copier was used on this copy routine and again the copying stopped at track 66 and would not proceed!



Several other options are available within the Copy menu: Make sub to make sub-directories; there is a Move button which will allow some selected files from the source device to the destination device and a Header button if you want your newly copied files to have the original date stamp and if you are using one floppy for copying files, you should set your source and destination drive with the same device name, such as flp1_.

The Sub-Directory menu is straight forward, just HIT it and another small window appears stating 'Enter name of new sub-directory: flp1_ (this can be any source device). Enter a name for your sub-directory and HIT it and it is done! The next option you have from the Files menu is Delete files. After you have HIT the Delete bar, a small window opens and there is a request, 'Are you really sure?', Yes or No. The pointer is automatically positioned on No! I have not tried to delete files with this option, it looks too OMINOUS for me, as all selected files will be deleted and once they are deleted, they are gone forever!

The next three commands have to do with string replacements, Rename replace, Rename erase and Rename add. Each of these functions will request string(s). The last command within the Files menu is Convert. Convert will allow the DM5 user to convert device names from one device to another, such as: Mdv to flp, and so on. You advise DM5 that you want to convert a string in a program. If you would like to change a word like 'even' to 'odd', you must add a space either before or after the word odd so that the byte count for the file is exactly the same as before a string was replaced.

At this time I believe that running through all of the remaining commands along the menu line will probably become tedious or uninteresting for our readers, so I will just OUTLINE the remaining commands through the menu line. This way you will know what is available within DM5.

The Options Menu is only available when a directory has loaded. Most commands from the Options menu deal with file selecting, with which the user can select files manually, or many other kinds of criteria.

Sort files: Files can be sorted by Name, Length, Type, Space and Update date and can be sorted in Descending or Ascending order. Choosing Unsorted will allow the directory to remain as the directory was loaded. Each file can be given priority, according to the Priority input. The default setting is '> E'. Note that there is a space between '> and E'. Sub-directories have the first priority, then follows ordinary files, executable files and last, relocatable files.

Change case: Any file name can be changed into Lower case or Upper case and or Mixed case. The user can configure which type of case you want for default.

Select by Name: This function has a powerful wild card system for file selecting. As an example, you can select Quill files which are appended with '_doc'. If you enter '*_doc' at the input line, every file ending with _doc will then be selected. The DM5 manual will show the user just how to use this powerful wild card system.

DISK MATE 5 - (cont'd)

Select by date: The Select by date command searches for files with the update date. The command brings up a window, displaying current date and time. You can adjust this to any date you wish and then HIT OK.

Select by number: The Select by number command offers the user for selection of files by File length, Data space, File version or Length of file name.

Select all sub-directories: Selects every sub-directory in the media directory.

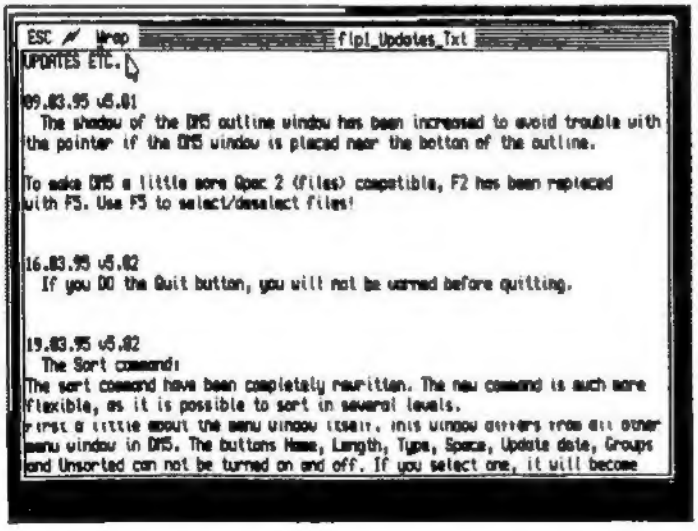
Search for: This command allows the user to enter a string and the QL will search for the contents of this string in every file in the media. The length of the string, Max length is used to limit the length for the search. The default is 0 and if you do not change this default, the whole file is scanned.

Deselect files and Deselect sub: Both commands does exactly the opposite of the Select commands. One deselects files and the other deselects sub-directories.

The next menu along the Menu line is Custom. This window provides information about your operating system (the info is based on my QL) as follows:

ESC	CUSTOM
SuperBASIC version	: JSL1
Processor	: 68000
QDOS Top of RAM	: 2048
Free memory, KB	: 1372
Free memory, bytes	: 1404928
Jobs running	: 2
Plug-in EPROM	: No
Screen mode	: Monitor
FileInfo loaded	: No
Printer port	: Ser1
Set time and date	: 01.12.96
Sort directory	: No
Change case	: No
Warn before action	: Yes

NOTE: The DM5 manual will explain the contents of this menu in detail. This info was gathered from my second QL set-up containing a Gold Card and Minerva ROM. As I have been writing this review, I have been using two QLs, one for writing this review (Super Gold card and Minerva ROM), and the other QL has DM5 running so that all information stated is correct and the operation of each command operates exactly as stated!

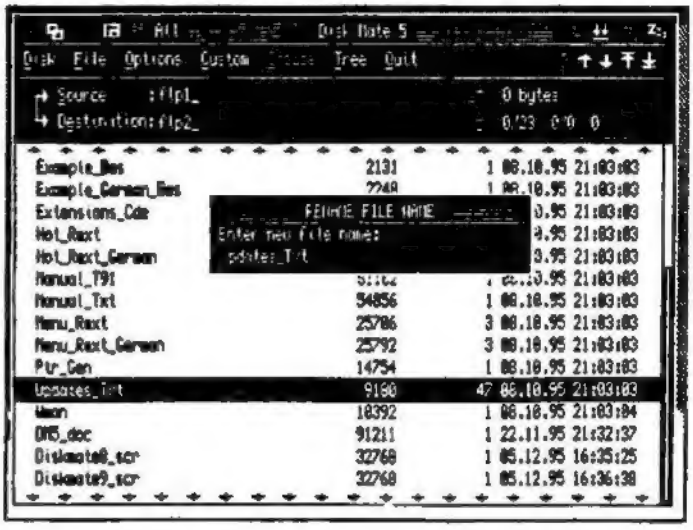


The next Menu along the Menu line is Groups: I could not activate the GROUPS menu. I opened the DM5 manual and looked up GROUPS for information pertaining to the menu. I could not find any info in the manual to help me, so I navigated the pointer to 'ALL', HIT it and the GROUPS menu came to life. I believe that this was an oversight on the author's part and in no way is this a criticism and perhaps, I may have overlooked something! The DM5 manual is better than most QL software manuals which I have read from other software authors.

DISK MATE 5 - (cont'd)

The command is available only after a menu has been loaded and has at least one file. You can make up to 256 different groups in a single directory. When you HIT the Group files bar, a window opens and presents the user with 256 characters. It is up to the user to select a character for the group of files you wish to sort. As you move the pointer down or to the right, the character appears on the bottom of the window along with the corresponding ASCII number.

When you have selected a character for grouping files, HIT it, navigate the pointer to ESC, HIT it. Now the directory should be sorted. HIT the Groups button from the Sort menu, and choose Ascending or Descending order.



Deselect Groups: It is my understanding that the Deselect groups button will make files group independent. I have not tried this command!

TREE: A tree contains every file on the media and you will not see any change to the directory if it does not contain any sub-directories. If you DO the Tree button, a directory will be loaded.

The next button is QUIT and this command should be self explanatory!

There is one feature of DM5 that I really like using the Question mark cursor. I activate the 'ALL' button with either a HIT or DO and then navigate the question mark cursor into the date and time column within the directory; select a file with a HIT, and then a window the size of the monitor screen opens up and displays the contents of the file. When you purchase new software, most users will look at the files directory for an Updates file or a _doc file or perhaps a _txt file. This method allows the user to view the contents of, say, a _doc file without having to load in Quill. A txt file can be copied to the SCReen if you wish, however you must use a scroll lock repeatedly or you loose a portion of the text. How many time do we, as software users load in the BOOT file into SuperBASIC to read the BOOT contents and if it is a large Boot program, again it scrolls quickly by. Read it within DM5!!!

You normally back-up any new software you purchase so why not use DM5 for making back-ups and then, search and look up any information files in DM5 after you have backed-up your new software purchase.

I have noticed that when loading a machine code or a compiled program, some of the code spreads out to several screens in length. There is a WRAP button at the top of the screen, however I can't get it to Wrap. Scrolling a file to the end of the file can be accomplished with HITting the pointer anywhere within the file window. ESC will return to the directory.

If you would like to RENAME a file, you can do this without accessing the Files menu. Place the question mark within the boundary of the Filename column, and HIT it. A window will appear requesting a Filename change. You can write a new filename if you like or if you want to leave this menu, DO it and you will be in the files directory.

I really have had a good time attempting to learn how to use DM5 and the Pointer Environment and I would like to thank Bob Dyl for allowing me to do just that! DM5 is an extremely powerful Disk and Hard Disk utility and the author, Pal Monstad should be complemented for his ability combining so many useful commands within DM5 and providing the user with a useful operating manual, which I used many times for help when I could not activate a command. As I stated earlier that the DM5 manual is better than most software manuals and this is due to the writers ability to express and write in simple terms. Thank you Pal!

Because of this exposure to the Pointer Environment, I will order QPAC II in early of 1996 and perhaps, I will purchase an additional Mouse Interface and Mouse from QUANTA.

sinclair Z88

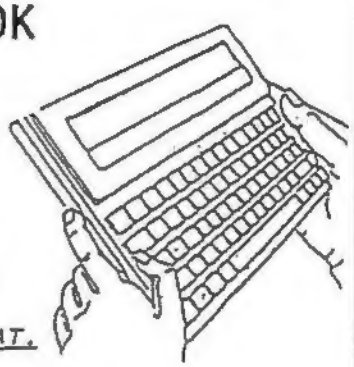
OFFICE/FAX 01494-871319 (EEC) W.N. Richardson & Co.

MOBILE: 0850 597650

6 Ravensmead
Chalfont-St-Peter
Buckinghamshire, SL9 0NB

THE IDEAL PORTABLE COMPANION FOR THE QL

THE CAMBRIDGE Z88 A4 NOTEBOOK



WITH BUILT-IN WORD PROCESSOR, SPREADSHEET,
DATABASE, BASIC, CALCULATOR, CLOCK, ALARM,
CALENDAR, & VT52 TERMINAL.
USES 4XAA ALKALINE CELLS(c.20 HOURS)

CAMBRIDGE Z88 A4 NOTEBOOK COMPUTER & ACCESSORIES

IT IS RECOMMENDED THAT ACCESSORIES ARE BOUGHT SOON AS THE PRESENT STOCK IS RUNNING LOW. NEW STOCK WILL HAVE TO BE REPRICED. *ALL UK PRICES INCLUDE VAT.*

	UK.	USA.
CAMBRIDGE Z88 COMPUTER.	£99	$130
2k RAMPACK	£16	$22
128k RAMPACK	£28	$37
512k RAMPACK	£60	$80
1Meg RAMPACK	£120	$160
32k EPROM PACK	£16	$22
128k EPROM PACK	£22	$30
256k EPROM PACK	£45	$60
EPROM ERASER	£32	$43
PARALLEL PRINTER LEAD	£22	$30
SERIAL PRINTER LEAD	£18	$24
MODEM	£90	$120
MAINS ADAPTER (230vac; 6v,500ma)	£15	$20
TOPPER (PROTECTIVE COVER)	£12	$16
CARRYING CASE	£16	$20
'Z88 MAGIC' A BETTER USER HANDBOOK	£15	$20
FILE TRANSFER TO OTHER COMPUTERS		
SPECIAL Z88-QL SERIAL LEAD	£10	$12
COPY DISKS OF QUANTA PROGS IMP/EXP & ARCHIVE EXPORT Each	£2	$2
PCLINK KIT (For PCs)	£25	$33
Z88 TO MAC KIT	£30	$40
Z88 TO BBC KIT	£15	$20

UK PRICES INCLUDE VAT; USA PRICES DO NOT. FOR OTHER NON-EEC COUNTRIES DEDUCT 10% FROM THE UK PRICES.
POSTAGE UK £5. EEC £15. USA £20. OTHER COUNTRIES £30.

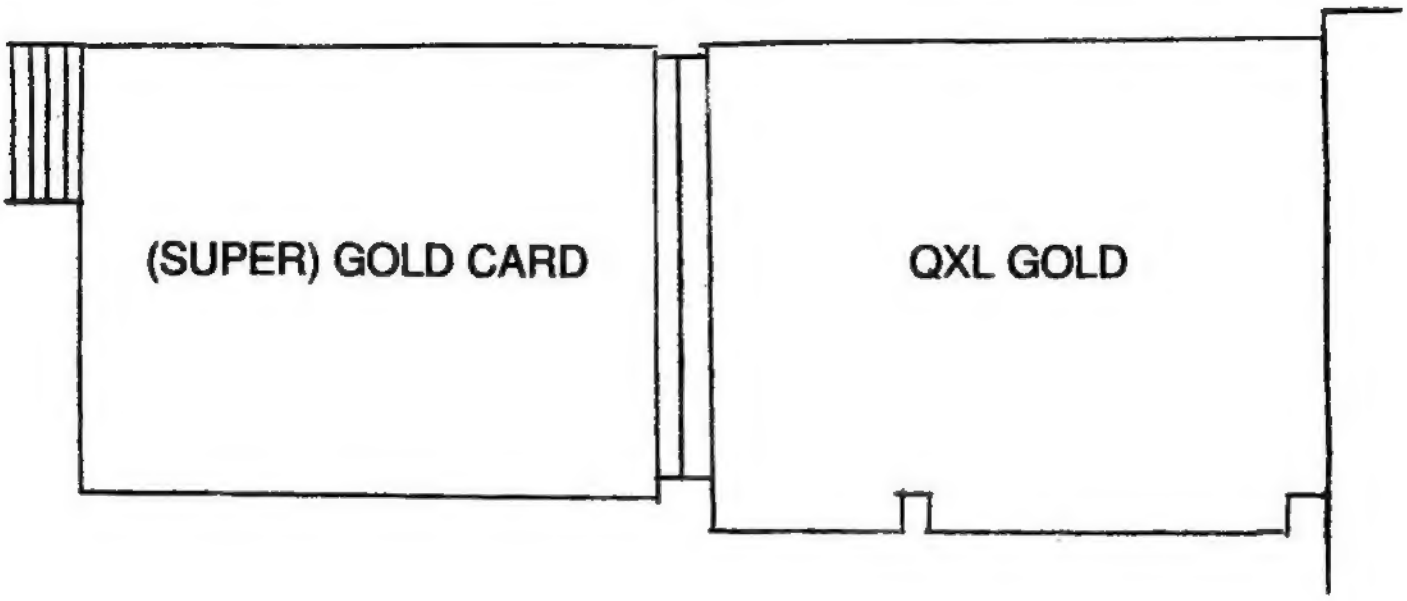
ALL THE STOCK IS NEW AND IS WARRANTIED FOR 90 DAYS. IN THE EVENT OF REPLACEMENT BEING AGREED, BUT THE ITEM BEING OUT OF STOCK AT THE TIME, A REFUND WILL BE MADE PROVIDED THE ITEM IS RECEIVED IN GOOD CONDITION.

QL & PC COMPUTER USERS WILL FIND THE CAMBRIDGE Z88 ESPECIALLY USEFUL FOR WORK AWAY FROM THE DESKTOP, WITH TRANSFER PROGS DATA CAN BE SAFELY EXCHANGED WITH THEIR DESKTOP SYSTEM.

W.N.RICHARDSON & CO CONTINUES TO PROVIDE FULL SPARES AND SERVICES FOR SINCLAIR COMPUTERS, QL, & THE CAMBRIDGE Z88.

MIRACLE SYSTEMS LTD

QXL GOLD



£100.00 fully inclusive (£90.00 outside EU)

* Alternative to the QXL
* Uses the processor and memory of (SUPER) GOLD CARD
* Low cost solution

The QXL GOLD is a low cost alternative to the QXL. It connects onto the 16 bit (AT) ISA bus of a PC and has a QL type expansion connector onto which can be plugged either a GOLD CARD or SUPER GOLD CARD.

The low cost is achieved by making use of the processor and memory of the (SUPER) GOLD CARD so the speed will depend on the type of GOLD CARD it is used with.

It is anticipated that delivery will commence towards the end of November 1995. Orders are being accepted now. Credit cards will not be charged and cheques will not be banked until the item is ready for despatch.

20 Mow Barton - Yate, Bristol - UK BS17 5NF

Tel/Fax: +44 (0) 1454 883602

Forget !

the QL keyboard membrane shortage.

Forget !

the QL keyboard problems.

Here !

for the QL, is a low cost, small, simple yet comprehensive keyboard interface

Note *This product is suitable for connection to most IBM AT style keyboards. Compatibility with other, older or multi-system keyboards is uncertain.*

Features

- Low Cost - only **£32.50**•
- Suits 101/102 key IBM AT style keyboards
- Keyboards available *(from £18.00)*
- Easy Fitting
- Most keys translated to QL formats
- Keypress record/playback facility
- External keyboard lock facility
- Small size

For the price of two keyboard membranes, an interface that will last a lifetime!

Prices

Interface	£32.50
Soft Touch keyboard	£18.00
Tactile keyboard	£24.00

Post/Shipping:

	Interface	+ keyboard
UK	£1.50	£4.00
Europe	£2.00	£5.00
Elsewhere	Enquire	

Amadeus Interlink

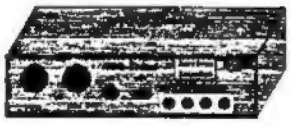
Share system resources, link computers, printers and real sound interfaces to a common network where QL's and PC's can talk to each other and share system resources. Up to a staggering 255 network interfaces can be connected.

More than one printer? no problem, any printer connected can be accessed by any linked computer. The multi-tasking QL for instance can effectively print to more than one printer at a time.

Transfer files between computers at high speed (*basic system software supplied contains a command 'AMACOPY' on both DOS and QDOS that enables file transfer between any linked machine).*

Sound - enhance your programmes to include verbal prompts and even musical interludes. Sound files are computer independent and may be transferred between any linked machine.

Straight forward, **low cost**, simple, fast networking from Di-Ren connects your QL to the Real World!



Ama-Sound Interface

Stop press - Amiga on-line soon

QL Network Prover

All time best seller from Di-Ren, this little box simply plugs in-line with your QL-QL network lead. An LED on the box indicates network operation thus keeping you informed of *what actually is happening!*. Only £4.00

Post/Shipping: UK £1.50, Elsewhere £2.00

Micro Process Controller (MPC)

Control any electrical appliance from this unit. It plugs in-line between your computer (or Amadeus interfaces) and a parallel printer. Two units can be connected in series if no printer attached

These units each house 6 make/break relays capable of handling DC and AC voltages of up to 240V AC @ 3 Amps and are easily controlled from software. Units are housed in a smart black ABS box, within which connections are made via fused screw terminals.

The controllers may be powered from a 9 Volt battery for very low usage applications. Alternatively a low cost 12 Volt DC unregulated PSU can be used. Output of the PSU should be at least 250mA for each unit attached. Suitable PSU's for UK use are available form Di-Ren

MPC Without PSU	£59.95
MPC with PSU (UK)	£65.50
UK PSU (500 mA)	£8.50
Postage UK	£3.00
Postage elsewhere	£4.00

Di-Ren

For further information contact us direct or visit our Internet site:
http://www.forthrt.com/~di-ren/products.html

Di-Ren
59 William St
Walsall
WS4 2AX, England

Tel/Fax +44 (0)1922 33580
Email 100736.1316@compuserve.com

Visa Access Mastercard Eurocard

QL-PC Fileserver II

Original Features retained

- ⇒ Connection to the PC's DOS Drives (including networks and CD ROMS)
- ⇒ Optional automatic conversion of Text Files for editing by either machine
- ⇒ DOS sub directory handling
- ⇒ DOS Read/Write attribute handling
- ⇒ Automatic recognition of native QL files (allows QL programmes to be Exec'd etc from DOS drives)
- ⇒ Works in background on QL & PC.

New/upgraded features

- ⇒ Access DOS devices, e.g. LPT Ports, Keyboard etc
- ⇒ Remote SCR and CON type text screen operation on PC display with colour, window and mode support
- ⇒ Up to 8 display screens on the PC can be operational and easily switched between from the QL.
- ⇒ Read PC screen data directly into the QL
- ⇒ Full QL filename lengths supported with options to rename drive names.
- ⇒ Advanced RS232 Comms handler for PC implemented as a DOS Device Driver (similar to QL SER device drivers)
- ⇒ Connection to PC via Serial links, Amadeus Interlink or any other suitable linking mechanism.

Price: £35.00

Upgrade from version 1 for just £7.00 + return of original masters.

Di-Ren Infolink newsflash

Amadeus System software may now be downloaded directly form our Internet Site.
Check out:
http://www.forthrt.com/~di-ren/amadeus.html

The World Wide Web

Kaub, GERMANY - Robert H Klein

Many of you will have heard of the mainstream computers new madness, the World Wide Web. This little article is not intended to introduce you into the World Wide Web, but just give you a little overview of some of its possibilities. I'll assume that you are connected to the 'net' and know how to handle the necessary programs. Articles regarding this can be found in almost every mainstream computer magazine. More specifically this text will give you addresses on the net where you can find people that are concerned about the fate of our beloved QL.

Let's take a look at the existing QL pages. There may be more by the time this is printed. How you can search for them is described below. The page of Di-Ren is at:

http://www.forthrt.com/~di-ren/homepage.html

The author of the QL Hacker's Journal, Timothy C. Swenson has pages that not only cover the QL but other Sinclair Stuff too. It's address is:

http://www.serve.com/swensont/

If you have followed all the links on these pages you'll have probably discovered links to other QL related stuff and perhaps even followed them. I'm currently doing the same, as I couldn't find my QL web pages list. I'm now on Peta's page, having followed the links to Di-Ren and Timothy Swenson. Now I'm going to follow the links to my own pages at :

http://www.uni-mainz.de/~kleir000/ql/
http://vzdmzi.zdv.uni-mainz.de/~kleir000/

Ye,s as you have noticed, they're on different web servers. At the second address you'll find not only the standard information but additionally some experimental web-ftp service. It's not like ftp, because you cannot connect directly to the site. But like a ftp server you'll find some QL software to download. Though there's only a limited diescpace, I've managed to put XChange and a current C68 release onto the page. Another thing I've stored there is my QL Service list which covers the addresses of all QL oriented vendors, QL clubs and a list of Internet resources. The list is made up in a way that allows you to click on specific internet addresses, be it ftp, mail or html and then it automatically invokes the requested action. (Now after I've mentioned the list I'm obliged to bring out an up to date version before this article is printed.)

Now, after you have browsed all this QL related stuff you're surely asking yourself, how you can find resources, be it QL related or not, on your own. On the Internet there are a lot of so called search engines, where you give some key words which should be contained in the title, header or text of any web pages, and after some time get a result. Depending in the given key words and the search engine you'll be given a list of links (or a 'no match' message, such a tough luck!), where you have to decide, which one is useful for you and which one is not. Some search engines provide a 'score', but it's you who decide, what's right. I have a web page at:

http://www.uni-mainz.de/~kleir000/such.html

which covers a lot of these search engines. If you don't want to search a search engine for other search engines take this page as a first start.

Now let's make an example for searching. Assume you are on the page with the links to the search engine and you are a great fan of the actress Nicholle Tom. (If you have followed all the links on my pages you'll have read this name before!) Let's take the search engine 'Inktomi'. Type 'Nicholle Tom' into the form field and press the 'search' button. Currently I get the result '26706 documents satisfied your query: Nicholle (650), Tom (26204) and a long list of links to the found web pages beginning with a 'clinical neurology home page'. Obviously that's not what I want. Back to the search form, only entering 'Nicholle'. This one is even more unusable. Now I know that Nicholle acts in the CBS comedy series 'The Nanny'. I decide to go the other way and to give as much as possible keywords: 'Nicholle Tom Nanny Beethoven Heather David' (Beethoven is another film she acts in, Heather and David are her siblings). That was a hit in the bulls eye, but they seem to have deleted the link to my fan page. Let's take another. Back to the search page and I choose Webcrawler this time. This one is better if you

give less keywords. Let's take it only with 'Nicholle'. One hit, my fan page. Doesn't help much if I'm searching for other resources.

Another try with setting 'find pages with any of these words' instead of 'all'. Again as much keywords as possible. First hit is my fan page, the rest seems to be unusable. Just what I expected. Too keep it short, you'll have to try a lot until you find the right measure how to give keywords to a search engine so that it gives you useful answers. Well, just another example: I tried all keywords (in lowercase) on the infoseek search engine and 6 out of the 10 first results are usable. At last I want to give you a very bad examples of keywords to use. If you're searching for information about computerized warfare it's a very bad idea to search for 'robot war' -- you'll get dozens of Starwars fan pages.

Though the last part of this article was not about the QL I hope it gave you a good example of how to use some of the web's features. If you have any questions you can reach me (where else?) on the net, my email addresses are:

kleir000@mzdmza.zdv.uni-mainz.de
kleir000@goofy.zdv.uni-mainz.de.

One last word concerning netnews. Some articles have been written about this in the past, but there was never that much feedback in the net. Many of you subscribed to the sinclair newsgroup comp.sys.sinclair, but soon unsubscribed because of the QL volume in the group being too low. You can't expect a high volume if you're not talking! For example there has been a nice discussion about QDOS running on a QL emulator on a Atari emulator on a Linux PC. Perhaps you are the one who suggests to try SMSQ on the Atari emulator. Come back and join us!

QL-THESAURUS

QL-THESAURUS is, as its name implies, an electronic thesaurus for the QL and QDOS compatible computers It is primarily designed to multitask with QL word processors to give the user fast on-screen help during a word processing session. Using this programme a writer can widen his choice of words or receive help in finding the elusive word on the tip of his tongue

QL-THESAURUS has a data base of 23,000 words and phrases divided into 1,000 different groups. It is heavily based on Roget's classic categorisation of words

To start a search all you have to do is type in the first few letters of the search word This accesses the dictionary or data base at the appropriate place. You can scroll through this dictionary to find the word you want or its nearest equivalent, and then press ENTER or SPACE to continue the search

This is a "Dictionary" search. If you want to look up a phrase instead of a word, you would use a "Word" search. In this mode you type in a complete word and all references to that word in the data base appear on the screen. You use the cursor keys to locate the word or phrase you wish to examine and then press ENTER or SPACE

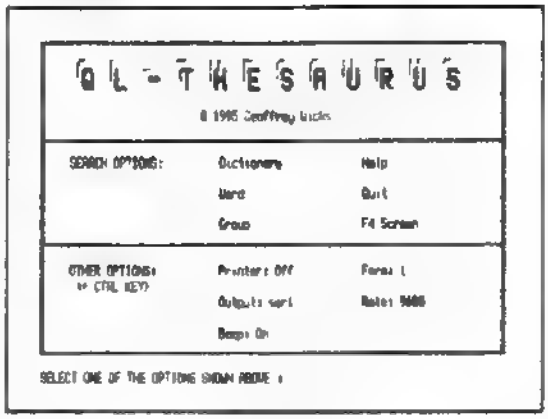
There is a third search mode, "Group", which is useful if you want to look for antonyms. In this mode you enter the number of the group you wish to search. A list of groups appears on the screen You can scroll through this list to choose the group you wish to view

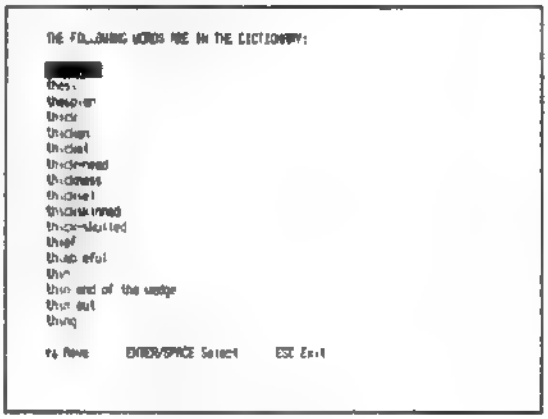
QL-THESAURUS makes extensive use of scroling menus to enable the user to work simply and quickly.

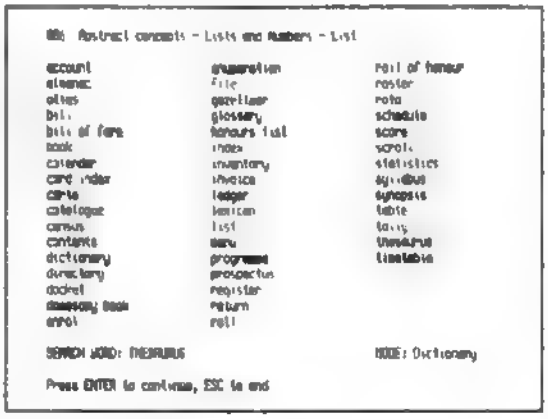
When you use QL-THESAURUS you will discover that there are two types of search, "Simple" and "Complex". Whether a search is simple or complex is not determined by the user, but by the nature of the search word "Brown" is a simple search. It has only one meaning (a colour) and appears in only one group of the thesaurus. "Maroon" is a complex search as it has three separate meanings (a colour, a warning signal and abandonment), and appears in 4 groups in the thesaurus.

In a simple search the words in the group appear straight away At the top of the screen is the number of the group, which is also the number in Roget's classification. Adjacent to this number are three key words or phrases which describe the idea conveyed by the group.

In a complex search a list of the groups in which the search word







appears is first displayed on the screen. You can scroll through this list using the cursor keys. The three key words or phrases will help you to choose the groups you wish to view. You mark these with the SPACE key When you have done this you can view the groups by pressing ENTER

Maybe this sounds complicated, but, to help you on your way, QL-THESAURUS comes with a 12 page, 4000 word manual complete with 6 screen illustrations. The manual has a 1 page "Quick Start" section, which will teach you how to use the programme in under half an hour

To use QL-THESAURUS you will need a QL or QDOS compatible system with disk drives and 640K memory. When fully loaded with its dictionary QL-THESAURUS needs about 490K free memory It would be difficult to multitask QL-THESAURUS with a word processor on anything less than a Trump Card

The dictionary of QL-THESAURUS, which is separate from the main programme is in a simple form and can be modified by the user provided he has a word processor or text editor that is suitable for this purpose.

Also available:

SOLVIT-PLUS 2

Electronic dictionary for crosswords and word puzzles.

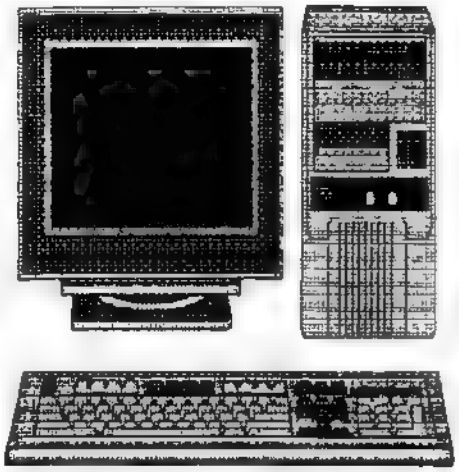
£15

QL-THESAURUS is available direct from the author and costs only £15. If purchased together with its sister programme, SOLVIT-PLUS 2, the cost is £25 If you pay by sterling cheque drawn on a UK bank you need have no worries about hidden currency charges. If you pay by Eurocheque, please pay in Dutch Guilders

£15 = $22 or 40 guilders

£25 = $36 or 65 guilders

Geoff Wicks, Bertrand Russellstraat 22, 1097 HL Amsterdam, Netherlands.
Tel. (31) 020 - 692 1521.
Netherlands Bank. Postbank number 4111942
UK bank: National Westminster East Ham Branch



Q.L. Mini Tower Kit

The QL Mini Tower Kit comprises of the following components -

1 - PC Mini Tower Case complete with 200 watt P S U
1 - QPlane powered back plane
1 - Sinclair QL Motherboard (JM or JS ROM Version).
1 - 8 pin DIN chassis socket (Monitor connection)
1 - 5 pin DIN chassis socket (Keyboard connection)
2 - 3 5mm jack sockets (QL Local Area Network connection)
1 - Di-Ren Keyboard I/Face + PC Keyboard

All the above fully fitted into the PC Mini Tower Case

£180.00p (JM Version) £190.00p (JS Version)

£20 PX for JM QL £30 PX for JS QL £32 PX Keyboard I/Face £18 PX Keyboard

PD & Shareware Software

Over 70 Disk's of Public Domain & Shareware Software for the Sinclair Q.L which include the following -

Psion Xchange V3.90L, C68 'C' Compiler V4 20, QL Emulator (AMIGA) V3.23, Molecular Graphics V5.12, LineDesign 2 Demo, MicroEmacs V3.11, Page Designer 3 Demo, Climes & lots more.

LineDesign 2 Clip-Art

Over 100mb of Adobe Illustrator Clip-Art files. These files will all load directly into LineDesign 2 Various themes are catered for such as Food, Backgrounds, Cartoons, Transport etc etc.

S.A.E. or I.R.C. for PD, Shareware & Clipart Catalogues.

QUBIDE
QL AT/IDE Interface

Allows you to connect modern AT/IDE Hard Drives to your QL. A massive amount of storage space can now be made available for your programs and files Compatible with SGC, GC, TC and most memory expansion systems, also Minerva & Hermes compatible

£65.00p

fully inclusive of P&P (UK)
+5% (Europe), +10% (Rest of World)

2nd User QL Hardware and Software

2nd User Hardware such as Colour and Green Screen Monitor's, Printers, Memory Expansion, Disk Drive Units, Disk I/Face's etc etc.

2nd User Commercial Software from DP, DJC, Jochen Merz, Talent, Eidersoft etc etc.

3 5in Floppy Disks DS/DD & DS/HD Pre-Formatted QL, Microdrive Cartridges, QL related Books and lots more.

AT/IDE Hard Drives

A selection of new and used AT/IDE Hard Drives compatible with QUBIDE available All Capacities from 40mb upto 850mb. Please ring or write for current stock availability.

QPLane

QPlane is a powered back plane for the QL, primarily designed to be used in conjunction with a PC Mini Tower Case and a PC PSU QPlane has 2 power connectors known as P8 and P9 which marry up with the special power connectors of the PC PSU to supply all the power requirements for your QL and expansion units. QPlane has 3 expansion slots.

£25.00p

Fully inclusive of P&P (UK)
+5% (Europe), +10% (Rest of World)

QUBBESoft P/D

38,Brunwin Road, Rayne, Braintree,
Essex. CM7 5BU. U.K.
Tel: +44 (0)1376 347852
Fax: +44 (0)1376 331267

META-DRIVERS

Zapresic, CROATIA - Zeljko (Nasta) Nastasic

Have you ever needed to do something like flp3_use win1_temporary_directory (instead of the more restricting win_use)? Did it ever seem nice to you to be able to use existing drivers to access new devices? Have you ever wished file name length limits did not interfere with network routing? Have you, for that matter, ever wanted to use any serial link (or otherwise) in the same way you would use the QL net? Even if the 'other end' was half the globe away? Well, if your answer was yes, to any of the above questions, this is for you.

IN THE BEST OF WORLDS..... all these problems would already have been solved. Having been in situations to ask those questions myself, I started thinking on how some of this could be implemented in QDOS/SMSQ.

There I was, minding my own business... One day, a few years ago, I got this wild idea about a universal network driver. This driver would be able to take a parameter telling it what physical device it is to use for networking, and it soon became apparent that it would be nice if the same parameters could be supplied to another driver on a remote machine. The big problem here was that the traditional way of doing this by use of commands simply could not work for a driver on a remote machine. What turned out to be a similar problem emerged again when trying to find a clean way to assign logical drive numbers to physical drives, or to partitions on physical drives or removable media. Then, all of a sudden (and in a most strange place) I had this idea...

Opening channels to drivers instead of devices... As I continued thinking about this, it soon became obvious that logical to physical mapping is only the start - the common denominator always turned out to be how to tell the driver what you want it to do, so in effect talk to the driver without accessing it's devices.

Arguably, the problem really has nothing to do with QDOS, but with the driver itself - all considered, QDOS should never need to know about internal workings of a driver as long as it supplies the necessary routines. However, the problem does come down to QDOS because it does not supply a clean way for jobs to talk to the drivers themselves, telling them how to configure the devices which are going to be used. In fact, it does not even recommend anything on this subject - so solutions used in various drivers are as many as there are drivers, it seems. Well, then, it all comes down to this:

(1) A way to cleanly send and receive driver configuration data.

Interestingly, adding only this into the current set of functions adds whole new possibilities, as I will try to demonstrate in this text. First of all, back to the basics. Each and every driver, including the very oldest, has to implement three basic functions:

(1) Logical to physical mapping: This is what determines which physical device is accessed when a device is referred to by it's name, for this purpose that would be the whole name, e.g. win1_something. Depending on the device driver, all of the parts of the name, i.e. win, 1, and 'something' could be significant in deciding what piece of hardware this really means. For some drivers and devices this decision process can be trivial, but for others it can be quite complicated.

(2) Protocol: This is what determines how data represented in QDOS looks when passed to the device, and the other way around. For instance, for a directory device this function handles how sectors (which are the basic 'unit' of storage) on the device are put together in files, directories, and indeed sectors as QDOS sees them (note that what QDOS sees as a sector could be different than the actual sector on a disc, case in point - ED drives). For network devices this usually implements a means to send commands and parameters to remote devices. For simple serial ports this could handle converting CRLF to CR. Also, it could do nothing - a good example again being a serial port with no handshaking - so this function can again range from very sophisticated to very trivial.

(3) Hardware access: This function implements means of actually talking to the hardware, to various memory locations etc, implementing additional protocols which relate to the internal workings of the hardware, rather than to the data that is transferred. This is always specific to the hardware in question.

What is common to all of them is that among apparently wildly different drivers and devices, some of these functions can be very similar. For instance, for all file bearing devices the way a file name denotes the actual place where the file has to go or come from is almost identical. Now back to the addition, which I will now propose as function (4):

(4) A way to cleanly send and receive driver configuration data: The purpose of this would be to use this configuration data to set up the way functions (1), (2) and (3) work. Because this ranges from fairly similar between drivers to wildly different, parameters have to be very flexible. This flexibility has also one very important, and not very obvious consequence:

In effect, given the right configurability, you can let one driver do Mapping, another do Protocol, while still another does Hardware access, or any other combination. Depending on what the drivers allow, this means you can have several logical devices share a common physical device, as well as a logical device share many physical devices. In effect, you can use certain parts of any driver to combine them into a driver that suits your needs. From this I got the idea for the name for the drivers implementing this principle:

Meta-drivers... Before I go into more detail, let us explore a way to implement the additions. QDOS divides IO devices into two loose groups:

1. Directory devices, which have the property of organised memory
2. Simple serial devices, which do not have this property.

What is really a third group, namely devices that, in addition to data, also transport operation requests to other devices for purpose of remote access to them, are put under the first category, by virtue of the way logical to physical mapping is done in them, something which I will illustrate further in the text. The first category is chosen because the second is a subset of it. For access to devices of both types, QDOS provides a set of basic functions:

1. Open a channel to a device
2. Close a channel
3. Input and output data to and from a channel
4. Format device
5. Delete file

The last two relate only to devices which can support the concept of files, those being the ones in the first category, the directory devices.

QDOS uses the concept of channels to do all input and output, thus presenting all devices in a similar manner. The device which is actually used is selected in the open call, and in format and delete, which themselves do not use the concept of channels. But, once a channel is open, the job using it does not need to know what hardware is being accessed or how, or even where - that's what the driver does.

QDOS also defines how a particular device is selected. This is done by passing the name of the device as part of the open parameters, in a string, which is in QDOS simply called a file name. In QDOS a file name also holds 'instructions' as to where the actual file is to be found.

There actually is a 'clean' way to implement the additional function (4) for any driver in QDOS - defining special 'devices' that represent the drivers themselves. Then, one only needs to open a channel to this new device, and send data to it, which it would then interpret and adjust the driver's workings accordingly.

Well, which device name do we use? Consider the following:

In QDOS device numbers have traditionally started at 1, and I see no reason to break with that particular tradition. In fact, it points to a way to implement opening a channel to a driver rather than to a device, which, when you think what it's supposed to do has pretty much sense in it - this special channel can be opened using device number 0. There is a way of implementing this in all versions of QDOS/SMSQ I know of, and without going into details, it has to do with the way device names are recognised. For instance, to add win0_ to device

win, we have to implement a serial device driver called win0 and a directory device called win. The directory device will take care of win1 to 8, while the serial device win0 will handle the special case of 0 as a drive number.

A small digression: It should be stressed that this 'drive number' should more accurately be referred to as the logical device number, as it does not really have to relate to different drives, or to any drives at all at that (network, ramdisc). Actually, it can be said that when 'using' a device in any way you are always referring to a logical device - QDOS really only knows about these, as it is completely irrelevant what the 'real' device is. Having this from the very start is one of the great features of QDOS.

The 'only' thing left to decide once you have a means of talking to a driver, is what to talk about and how. More of this towards the end.

A taste of how it could look like... To see what news all this brings, I will attempt to provide several (hopefully clear) examples.

(1) Hard discs - One of the simpler uses that come to mind here are assigning partitions to drive numbers. For instance, let us say that you have an interface capable of having several physical drives, which can also have partitions. Then you have to solve:

- how partitions translate into drive numbers
- what if there are more partitions than there can be devices, or more than are needed at one time
- what if we need direct sector access to a physical drive in whole and not to a partition, for instance if we want to set up partitions in the first place or access non-standard formats or drives

This (and more - later) can be handled by being able to change logical to physical mapping. Let us say we want to set up two partitions on drive 1 as win1_ and win2_, and further access drive 2, which is as yet unformatted, using direct access, so we can set it up. Then, if win was a meta-driver (see last section):

```
OPEN #3,win0_1          [open channel to driver for setting win1_]
PRINT #3,'1,1'          [tell it to set win1_ as partition 1 on drive 1]
CLOSE #3                [not really needed, reopening closes previous ch.]
OPEN #3,win0_2          [open channel to driver for setting win2_]
PRINT #3,'1,2'          [tell it to set win2_ as partition 2 on drive 1]
CLOSE #3
OPEN #3,win0_3          [open channel to driver for setting win3_]
PRINT #3,'2'            [set it up as drive 2, direct sector access]
CLOSE #3
```

Although this is a perfectly valid approach, perhaps a better one would be to implement a basic procedure ASSIGN and a function FASSIGN$ to do the work. For instance:

```
ASSIGN '1,1' TO win1
```

This could actually transform win1 to win0_1 and open that channel, send it a string containing '1,1', check for errors, close the channel and return. Similarly:

```
a$ = FASSIGN$ ('1,1' TO win1)
```

This does the same but also returns response read from win0_1 (there is a proposal to make this standard, see recommendations at the end). You will notice that the syntax of ASSIGN and FASSIGN$ is inverse with respect to the example above. Although it is altogether not important if a logical device is assigned to a physical one, or the other way around, it is beneficial for ASSIGN and FASSIGN$ to have a reverse arrangement of parameters, so that linked assignments are possible using a TO...TO...TO construct and a bit cleverer code, but more of this later. Let us just assume that ASSIGN and FASSIGN$ are implemented. Then the above example becomes:

```
ASSIGN '1,1' TO win1
ASSIGN '1,2' TO win2
ASSIGN '2' TO win3
```

At this point you might ask what is wrong with the already existing WIN_DRIVE which does the same job, more or less, on Qubide or on the Atari under SMSQ - well, for one, ASSIGN and FASSIGN$ would work equally well with any other meta-driver. Also consider other possible problems:

- what if this particular interface is not the only one with hard discs or related devices on this system
- what if we want to access remote drives without thinking where they really are
- what if we want other devices to appear as Winchesters for purposes of compatibility

Traditionally, hard discs have been referred to as 'win' in QDOS (with the exception of a very few of them!). The problem here is that win means 'Winchester', and that is a broad category, including all sorts of them, even the ones on remote machines.

Let us explore a hypothetical implementation of a winchester meta-driver win, with existing meta-drivers for IDE, SCSI and even the good old MFM. To avoid name conflicts, let us say that those were named ide, scs and mfm. The winchester driver win can be used to access all winchester-like devices, connected on ide, scs or mfm, or basically anything.

The driver win really only does little more than redirection, as set up by opening a channel to assign logical to physical mapping. As win actually knows of no physical devices itself, it uses other logical devices instead. Because of this, the string holding assignment parameters is actually used to open channels to other meta-drivers. Again an example using ASSIGN:

```
ASSIGN ide1_ TO win1
```

The win driver simply uses the supplied string before anything that will follow win1_ in any future access. Another example:

```
a$=FASSIGN$ ('1,1' TO ide0)    [this returns which ide device number was assigned in first character, see
                                recommendations at the end]
ASSIGN 'ide'&a$(1)&'_' TO win1
```

Here, win1_ is assigned to the first unassigned ide device, which in turn is assigned to partition 1 on drive 1 on the ide interface.

Making the assign commands a bit more clever in interpreting the assignment parameters could make it do linked assignments automatically. Then a construction like:

```
ASSIGN '1,1' TO ide0 TO win1
```

would do the same thing as the two above.

Using this it would be extremely simple to assign partitions on drives connected to IDE, SCSI, MFM controllers, and a network drive or two to win.

An interesting side effect: a driver like win does not use additional (limited) spaces in the QDOS definition blocks, as it sorts out which device should really be accessed sooner than the blocks are needed. Along the lines of the previous example this could be accomplished by:

```
ASSIGN '1,1' TO ide0 TO win1
ASSIGN '1,1' TO scs0 TO win2
ASSIGN '1,1' TO mfm0 TO win3
```

As you can see, this concept has very interesting properties - it can use all related devices regardless of the

hardware, using the same QDOS device name. However, things like:

ASSIGN win1 TO win2
ASSIGN win2 TO win1

or even worse:

ASSIGN win1 to win1

should be trapped and return errors, and in fact there is a way to detect such circular assignments.

The 'generic' hardware devices (ide, scs, mfm) can also be extended to handle assignments to other devices if needed, if the assignment string given is a name of a device. Arguably, even sub-directories on one device could be assigned to another device, and also files to serial devices. Some pretty wild things could also be done involving pipes.

(2) Protocol translation devices - An example of a front-end device which uses hardware access by another driver to implement a different filing system is a hypothetical implementation of a CD ROM meta-driver. Although somewhat similar in intention, the CD ROM meta-driver would be more complicated than the win driver described above, because it must also implement the Protocol part. It could use other drivers capable of accessing the hardware of a CD ROM, but not being able to interpret the file structure, by using only the direct sector access function of that driver, and building onto it .

Which meta-driver this would use would be regulated by assigning another device to the CD ROM device. In this way, if you have SCSI and IDE (each with it's own meta-driver) you could easily use the same CD ROM meta-driver to access a SCSI and an IDE CD ROM! Again an example:

ASSIGN '3' TO ide0 TO cdr1 [CD drive is IDE drive 3]

This would assign IDE drive 3 to be used by the CD ROM driver cdr to translate direct sector accesses to IDE drive 3 into something understandable to QDOS. However, that is not all by far. Guess what this would do:

ASSIGN '3' TO ide0 TO cdr0 TO win3

Given win, cdr, and ide drivers as described above, this quite neatly makes a CD ROM which is drive 3 on the IDE interface, use a CD ROM filter device cdr to appear to QDOS/SMSQ as win3_. What if the CD ROM is remote on n2? Well, no big problem there:

ASSIGN '3' TO n2_ide0 TO cdr0 TO win3

With two assigns the remote machine could be made to handle CD file structure conversion, a sort of primitive parallel processing. And - it does not even have any more problems with the name length limit than any normal network access. But more about networks later.

Note that cdr uses the assignment parameter differently than in the win driver described above - it does not call assignment of a logical drive in ide, but opens a file on the supplied device (whichever that ends up to be) for direct sector access to implement CD ROM to QDOS format conversion.

Another example would be PC disc IO - simply do a meta-driver which implements PC file storage routines using direct sector IO. Unfortunately it's not very up-to-date now that level 3 drivers exist in SMSQ/E, but its a good example.

In addition, _USE commands as used for changing names of devices (remember what that is intended to accomplish - NOT having exotic names, but having names of other devices for compatibility purposes) may become redundant. This is because:

WIN_USE flp - gets replaced by (if flp is clever enough): ASSIGN win1 TO flp1

and that leaves flp2_ to flp8_ accessible (whatever they are, if anything), as well as all wins including win1_.

(3) Networking - Expanding the same logic to other drivers, it soon becomes clear that there are some staggering possibilities, most of which come out in network driver design.

Let us again consider a hypothetical network driver, working similarly to the cdr driver mentioned above. Unlike the network driver we're familiar with, this network driver does not drive hardware, but rather implements a network protocol over a serial device. Which serial device is used? That would depend on (you guessed it) what parameters are given in the assign call. Consider the following:

r - an universal net protocol driver
net - companion driver for the standard net port, netxx makes connection to station xx

ASSIGN net TO r1

This sets up net as a means to access other drivers, using a networking protocol implemented by the r driver, in form of a device called r1:

DIR r1_10_flp1_

does dir flp1_ on net station 10 (actually doing dir r1_net10_flp1_), provided there is a server job there capable of interpreting the protocol. In a way this is very similar to what DIR n10_flp1_ would do. What's new is:

ASSIGN ser1 TO r1

will gladly use ser1 for talking to a remote machine, using the same protocol - or any other bi-directional serial device you care to name. Don't let me even start explaining what would happen if there is a modem connected to the serial port! But that is not all. Consider the following example:

ASSIGN ser1 TO r1
a$= ASSIGN$ net10 TO r1_r0 (with 1 returned)

In this case, the second assign looks for a free net device on the other end of ser1, and uses it to access an even remoter station 10, using the first-level remote machine on ser1 as a mid point. What's even better, if there are free net devices on this mid-point gateway, it needs to know nothing about that, all is handled by the assign routine on the first machine - note that all assignments have been made remotely from the first machine, in effect implementing network routing on the fly. Also, nobody is stopping you to do a further:

ASSIGN something TO r1_r1_r0

But what about the name length limit? Well, there is a way of solving this, by making the servers on the remote machines keep track of routing, instead of supplying routing information in the names themselves. This would easily be done by adding special routing information in the assign call. This way routing information does not have to be part of the name, which frees precious characters for other things. For this a special character in the assign string could be used to distinguish between the medium used for access and a prefix that is added to the filename once it is received on the remote machine, something like:

ASSIGN 'flp1_@ser1_' TO r1

Then flp1_ at the other end of ser1 (note the use of the @ (at) character, in the spirit of networking) is automatically accessed like this:

DIR r1_

becomes on the remote machine:

DIR flp1_

Note that the name length limit would never been exceeded, no matter how many levels of remoteness are used, because the final name is reconstructed only on the final machine. The routing information given in the assign

call is actually sent to the meta-driver as a string, so the maximum length can be up to 32766 characters - and you can route half the world with that! This simply can't be done by a NFS_USE command!

Having given the r driver a way to embed routing information into it's protocol, without the need to put it into a file name can be considerably expanded to implement peer-to-peer routing. For this to be possible the network server job has to be quite clever and be able not only to serve, but also to request service from other machines in the net independently, based on the received routing information. The routing information can be given to the r driver by extending the assignment parameters further:

ASSIGN 'target@transport1@transport2@transport3...' TO rn

The idea is that anything before the first @ is the real target device, and the collection of @-separated devices are routes signifying the location of the target device. Once this location is reached, the target device name gets the file name appended onto it and the requested operation is done with this new file name.

(4) Boot devices - Finally, to illustrate yet another use of meta-driver capability, a possible solution to the booting problem:

When initialised, QDOS first opens a serial device BOOT and treats the input stream as a basic boot program. If there is no such device, it opens a file called mdv1_boot. Traditionally, all drivers that are used to access devices that support booting from them use either one of two principles to enable booting:

(1) Implementing a serial driver called 'boot'. As this links in later than previously linked drivers, it is called first. Therefore, the last device initialised is booted from.

(2) Initially the driver is linked in with the name 'mdv'. Again, the last one linked in is the one that boots the machine.

There are two flaws with these methods:

1. Writers are usually inconsiderate and don't provide for the possibility that there might be other bootable devices which have to be called if boot fails. Other writers try to mend flaws in existing boot schemes in their own code with mixed success. For method one, when a boot driver is called, and fails to boot, it should change it's name (dirty - involves rummaging around system tables in hope of finding a matching name - the problem is that there could be many of them), and call the next boot driver.

2. Drivers that boot as mdv can cause problems. Usually the booting itself re-names the driver to it's normal name (with possible consequences as per 1), so if it was not the one doing the boot, it stays named 'mdv'.

The following gives a solution: A boot meta-driver is implemented, and the meta-driver's assignment facility is used to pass booting instructions. Instead of linking in additional devices or renaming themselves, further drivers for bootable devices simply assign a boot file to the device 'boot':

ASSIGN flp1_boot TO boot0

If there is another bootable device, eg. win, then:

ASSIGN win1_boot TO boot0

would put the file win1_boot as a booting candidate on a booting file list. The boot driver's assign routine simply records all assignments, i.e. all potential sources of a boot program. When the boot device is called, it tries to open any files that are recorded by the boot assign routine. The first one that succeeds automatically ends further booting attempts and clears the record. In this way a 'clean' way of attempting boot from any number of devices is possible.

(5) ...even more daring ideas - Now lets overdo it a bit suppose we have altered our operating system to support multiprocessing by delegating jobs to be executed to other computers accessible via various means (with a strong emphasis on meta-drivers being used for this!). In order to do this, job schedulers on different machines would need their own meta-device to communicate with one another, in order to negotiate which actual machine has to execute the job. Note that all devices the job actually uses can be on another machine - redirected without the knowledge of the job by communicating with meta-drivers. What would this mean?

Well, given the meta-driver ability to mask the actual physical device or it's location from the job using it, the job is free to be executed on any machine while the user is interacting with it from yet another machine.

Also, given the ability of dynamic network routing, any machine can search the net and find another machine, and then negotiate with it to have jobs executed on it.

This would mean that something could occur which, for no better term, could be called wide-area parallel processing. If the jobs in question can be moved in memory while being executed (much like the SBASIC jobs or the superBasic interpreter) there is nothing to stop one machine to toss a job to another in mid-execution, in order to free some of it's resources. Revolutionary? Well, I don't see Microsoft doing it!

Final word: Any comments and ideas are, as usual, very welcome. My thanks for all people who helped in defining the problem, sieving the ideas, and preparing this document. Also, thanks to the readers who have managed to get to the end of this long text, but I feel that the subject merits the length.

Addendum for the technical: Amongst the potential benefits of meta-drivers is that the means of passing data to a driver and getting responses, or status, becomes a system-wide resource, and can be used in a similar way for all meta-drivers. In fact, one could probably construct a nice pointer-driven program to configure all the devices attached to the machine. Granted, the nature of the data that is exchanged can be very specific, but also very similar - what is needed is to somehow bring the similarities together in a form of a 'standard'.

Here is what I have come up with until now:

(1) Meta-drivers can be of both categories, directory as well as simple serial.

(2) When the driver is accessed as a device, it behaves as a simple serial device. This only means it supports only simple calls, i.e. it does not support formatting, deleting or file pointer operations.

(3) The name supplied determines what driver function's operation is to be affected, namely function (1) - Logical to physical assignment - or functions (2) and (3) - Protocol and Hardware access. The reason this particular division was made is that changes to Logical to physical mapping are similar between devices, while the other two categories are more specific, and in some cases unchangeable (mostly in function (3)).
When opening a channel with a name of this sort, standard error codes are returned if the driver cannot understand or do what is asked of it.

(4) Any parameters that select what particulars should be changed, and how, are written in a form of a string into the channel. Writing unknown, invalid or too long parameters returns the 'usual' errors, an no operation is performed.

(5) Any response or status is obtained by reading strings from the channel. Before anything is read a selection of what is needed is done by writing. After a write, reading will return a response string or strings. Any read before a write or after the reading of the response string(s) should return the usual error, that being 'end of file'.

(6) It is recommended that changes requested become valid after the command/parameter string has been written, and before the response string(s) are read.

(7) No recommendations are made as to how many channels can be opened to a driver at once, or whether it can be done while files are open on the logical devices controlled by the driver, or if files can be open while the

driver channel is open. This is left to the discretion of the driver writer, as long as he takes care of possible consequences. In any case, QDOS/SMSQ already provides a mechanism that prevent attempts to open more channels to a device that cannot support that - this being the 'in use; error.

(8) Care must be taken to close the channel to the driver as soon as the desired actions have been performed, because of the consequences of the previous paragraph. The channel can be closed with response strings pending, which will automatically clear them.

(9) The following is the proposed standard for encoding names:

OPEN dev0_ - reads or sets options and configurations for Protocol and Hardware access.
OPEN dev0_n - n is the logical device number (1 to 8), sets or reads Logical to physical mapping for device dev1_ to dev8_.

OPEN dev0_0 - sets Logical to physical mapping of the first un-mapped logical device dev, returns devices that are mapped. Other constructions are considered driver-specific.

(10) The following are the proposed standard requests:

'?' - return status
'name' - set mapping to 'name' or return value of 'name', driver specific.

(11) The following are the proposed standard responses:

'' (empty string' - operation complete or waiting for write
'#number' - <number> of response strings follow
other - single response string per se, driver specific.

(12) additional propositions on requests and results:

(a) Assigning logical device n is done by writing a driver specific string to a channel opened to dev0_n. Response is a string in the form of 'nx' where n is the logical number of the device the assignment has been made to, and x has a bit set for each logical device which is currently assigned. Assigning a device that has already been assigned results in re-assignment with the new parameters if the device was not in use. When 0 is returned as n, no assignment was made.

(b) Finding out the assignment of a particular logical device is done by writing '?' into a channel open to dev0_n. The response will be the original string given upon assignment, or '0x' where x has a bit set for each assigned drive if the drive requested was 0 or un-assigned. Therefore an assignment string should not have the same form ('0x').

(c) Un-assigning a logical device is done in the same way as assigning but with an empty string given as parameter. The return string reflects that the device has been un-assigned, as stated above, returning '0x'.

(d) Writing '?' to a channel opened to dev0_ is recommended to return possible request strings for setting or reading device specific data, in the format of '#number' followed by <number> strings containing the request characters themselves. Capitals denote characters significant in the request string and lower case are wildcards which are disregarded (like in DEFine PROCedure). Any additional parameters in the request string or the format of the responses is driver specific. Perhaps an agreement could be made as to a limited set of standard simple requests (for instance Name, Version, Configuration).

The purpose of these is to provide a simple fixed set of requirements, so that basic keywords and utilities can be defined for common tasks, for instance an ASSIGN procedure, a FASSIGN$ function, and also functions to find out the assignment status of devices, etc.

WHAT WILL THE QL COLLECTION COST ME?

Just **£179 in total**. There is nothing to add, no hidden taxes, and P&P to anywhere on earth is included (add £5 for Airmail). You save over £2,100. DP recognises that you probably have some titles already (though perhaps not the latest releases), and some may not be of interest to you yet (likely to change when you see them!) - the price reflects this! **THE QL COLLECTION** is worth it even if you only want to update all your existing DP products: however, DP will continue to accept orders for *individual* DP programs at the prices quoted if you really insist.

WHAT EXACTLY IS INCLUDED IN THE QL COLLECTION?

You get the fullest, very latest, most up-to-date releases of all - **every single one** - of the **66** QL programs listed. The software is the finest, and the QL's very best. The titles would cost you over £2,300 (plus applicable P&P) to buy individually - you can check this by summing the prices overleaf, or those quoted in earlier ads. The only titles omitted are Mega Dictionary (only for 2Mb RAM systems - add £15 for it), MS-DOS v6.22 upgrade (add £90) and any less capable variant of a title that is itself included in **THE QL COLLECTION** (e.g., since the top-of-the-range PROFESSIONAL PUBLISHER is included, DESKTOP PUBLISHER is obviously excluded). You get both versions of PC CONQUEROR (as the list shows) so as to cater for all hardware variations. You may never ever need to buy another QL program again. The range of software you will get is truly staggering. It is too good to be true. But it is true - while the offer lasts....

WHAT ABOUT THE PROGRAM DOCUMENTATION?

All the latest applicable documentation (lots and lots of it) is included on disk, and can be read and printed using Perfection Special Edition or Editor Special Edition, which are also both included, and which can - of course - be used to search, browse, analyse or "edit" manuals at your leisure. Printed copies may be bought later if wanted - full details are sent with the order.

WHY CAN'T I FIND THE CATCH IN ALL OF THIS?

Because there isn't any. DP, whose QL commitment continues, makes this super offer to celebrate the birth of a marvellous baby daughter Michelle, now through all her early problems, to Julie and Freddy. **THE QL COLLECTION** is licensed for use by the purchaser alone, who by buying it agrees not to resell or otherwise pass on any part of it, or of any DP software already possessed. Technical support is negotiable: full details are supplied with the order for you to take up should you want to do so. DP *reserves the right* to withdraw **THE QL COLLECTION** offer at any time later than 14 days after your receiving this magazine, so please do hurry.

WHAT HARDWARE WILL I NEED TO RUN THINGS?

You will need a twin disk drive (DD, HD or ED, 3.5" or 5.25"), lots (123?) of blank disks and over 1.5Mb RAM (Gold Card, Super Gold Card, QXL, ST/QL and equivalents) to fully use **all** the software. The vast majority of titles will, however, work on much smaller systems: earlier DP ads indicate with precision the minimum hardware needed to run each program. If no disk size is specified when ordering, DP will assume 3.5" DD. If you do not yet have a powerful enough QL system, you may wish to contact a hardware dealer and buy, say, a second-hand Gold Card and/or twin disk drive (preferably 3.5" DD or HD - avoid Mitsubishi, and if HD or ED, ensure 100% compatibility with all Gold Cards is *fully guaranteed* by the supplier) as needed.

HOW CAN I GET MY COPY OF THE QL COLLECTION?

THE QL COLLECTION can only be obtained directly, by posting your order (including payment of £179 by cheque drawn on a UK bank / building society, Eurocheque or postal order, or quoting a VISA or MASTERCARD credit card no: and card expiry date) as soon as possible to:

DIGITAL PRECISION LTD, 222 THE AVENUE, LONDON E4 9SE

THE QL COLLECTION

QUANTA

Independent QL Users Group

The Largest Computer Club in Europe.
Now in our 12th Successful Year.
Worldwide Membership.

Formed in 1984, QUANTA (The QL Users ANd Tinkerers Association) has endeavoured to promote the Sinclair QL Computer and more recently its many offsprings but essentially the "QDOS" operating system as devised by Tony Tebby.

There is a large and growing, sophisticated, supply of software which seeks to take advantage of the many benefits offered by QDOS, SMSQ etc., such as Multitasking, a recent arrival on the PC but a part of QDOS for over 10 years, the Pointer Environment and the many advantages of 32 bit computing.

It is the perfect environment for the "Hobbyist" Computer User who will recognise immediately the many advantages once he has been introduced to Quanta and it doesn't matter whether he is biased in favour of Software or Hardware, the scope is enormous.

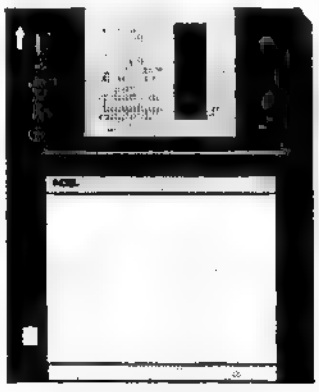
QUANTA maintains a library of 80 plus disks, mostly full, which is free to members and still growing. We also run "Workshops" so that members can meet one another and a great time is had by all. Pherhaps the greatest achievement QUANTA can boast about is the ease with which you can make friends and obtain help.

To misquote Isaac Newton, "If we have seen anything it is by standing on the shoulders of others"

UK Members £14.00 Overseas Members £17.00

Payment, in Sterling, by cheque drawn on a UK Bank, Money Order or Credit Card.

Send to our Membership Secretary:



Bill Newell
213 Manor Road
Benfleet
Essex
SS7 4JD
Tel. (01268) 754407

PROGS

Professional & Graphical Software

Haachtstraat 92, 3020 Veltem, Belgium, tel/fax : +32-16/ 48 89 52

ProWesS pre-release version available now

After more than a year's work, we are very proud that we can announce our new product, which we believe will be an important new environment for the QL.

ProWesS is the "PROGS Window Manager". It contains the complete environment for running ProWesS applications. This is currently the pre-release version. We do not consider the package to be finished just yet, however, especially the ProWesS reader which is part of the package may be interesting to many users, and we would like to get comments about the system as soon as possible.

The ProWesS reader is a major part of the package. It is a hypertext document browser. This means that text files which include formatting commands and possibly links to other files can be displayed and read in this program. This is used in ProWesS to read (and possibly print) the manuals, and display the help files. The hypertext documents which are used by the ProWesS reader are in HTML format, the format which is popular on Internet to display World Wide Web pages.

All registered ProWesS users will get a free update to ProWesS when the full version is available. The package currently contains (apart from the libraries) the ProWesS reader, which allows you to browse hypertext documents (in HTML format), the ProWesS loader, which allows loading applications, including all the required extensions without reset, and some small sample applications (like a calculator). Many more utilities and installation software will be sent to you as the free upgrade to the full version !

ProWesS does not include the programming documentation. These should be available via bulletin board and public domain software suppliers by the end of February. The programming documentation will be readable in the ProWesS reader, and partly in DATAdesign (the demo version will be included).

order your copy of ProWesS today ! for only BEF 2400

Payment terms :

ProWesS is available NOW for BEF 2400 (HD, excluding postage). It is normally distributed on high density (HD) disks. However it can be obtained on double density (DD) disks at an extra costs of BEF 100

If you are VAT registered (specify registration number) or live outside the EEC, the amount to be paid is the total (including postage) divided by 1.21 (no need to pay too much).

Payment can be done by EuroCheque in BEF, or by VISA, EuroCard or MasterCard. Credit card orders can be handled by phone. For credit card, please specify name of card owner, card number and expiry date.

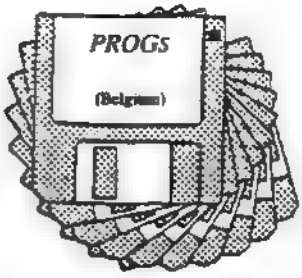
Postage : Costs of postage and packaging have to be added. You can choose the quality. Rate depends on no of programs.

	priority mail			*ordinary mail*		
copies	Belgium	Europe	World	Belgium	Europe	World
one	100	200	230	100	120	135
two	110	340	420	110	180	215
3 or 4	120	560	770	120	300	370
5 to 8	160	870	1250	160	550	675
more	200	1130	1610	200	800	1005

All prices are in BEF, including 21% VAT

ProWesS : (QL Windowing Meets the Future)

Sunnyvale, California, USA - James D Hunkins



The good people at PROGs Professional and Graphical Software are at it again. We all know them from programs such as Line Design and PFList. They developed the Proforma graphics engine which brought modern vectored graphics to the QL.

Now they are attempting to modernise the windowing environment. Building on a solid base laid by many other QL developers, they have updated their own Proforma package and added a new own window manager (ProWesS) and 'C' library routines (Syslib). Additionally, they have introduced a DLL Manager (Dynamic Link Library Manager) which, like standard QL 'things', allows a group of library routines to be shared by all jobs running on your system, saving resources by eliminating redundant code.

Articles have already been written on this package by the authors as they were developing it. However, now that ProWesS is becoming a reality, I feel that a first person review of it both as an end user and software developer is appropriate.

This particular article is just one of a series. In this issue we will have a brief first look from the end user's view point. Future articles will pursue the use of ProWesS from the software developer's stand point. They will be formatted as a simple tutorial/introduction to writing programs with the ProWesS package.

For this article I have been using the Alpha release of ProWesS. As expected, some bugs do exist (hence, the term Alpha release). The next issue will be based on the pre-release version which can now be ordered direct from Progs (see the advertisement elsewhere in this issue). I understand that most of the bugs have been fixed and more of the package has been completed. Within the next few months, the full release package should become available.

What it is: ProWesS can be considered a software support package. By itself, it won't do much for you. However, any program written to use it requires that you have it on your system. What it does for the programs written to use it is another story all together.

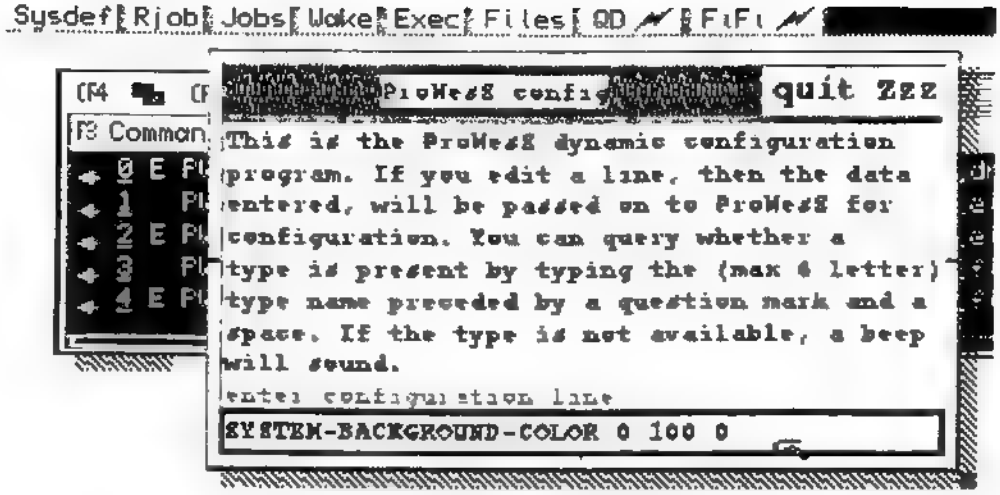
Currently with the traditional QL's window manager (Wman), programs are restricted to the built-in font sizes. Each program must be configured individually. And when you change screen size/resolution, the image gets smaller or larger accordingly.

With ProWesS, you have one central configuration file that configures all programs using ProWesS. This introduces a common look and feel to your system. This means less fiddling around setting up programs. As new programs will perform similar to ones already on your system, they will be easier to get proficient at using.

ProWesS uses an updated PROGS Proforma package for vector fonts and color graphics. Therefore programs will be able to use a wide variety of fonts at any size you wish. If you have a problem seeing the standard size fonts, make them larger. If you prefer to pack more information on the screen, make them smaller. You can even adjust font sizes for different areas of a program, such as the title block or input edit lines.

Lets say that you change screen resolutions or get new hardware that offers more resolutions and/or colors. You will not have to update your programs. If your current version of ProWesS supports the resolutions, the change will be transparent (except things will be sharper) unless you choose to change settings. If the hardware requires changes in the software, just get an updated ProWesS copy. Your software packages will 'inherit' all the necessary changes directly from ProWesS.

You also get a new look (see figure 1). For example, instead of the traditional icons inside a window for moving or resizing, windows will have a re-scaling border. When the mouse is positioned over the border, the cursor becomes a double sided arrow. By pressing the left mouse button, you get the familiar move icon. The right mouse button delivers the resize icon.

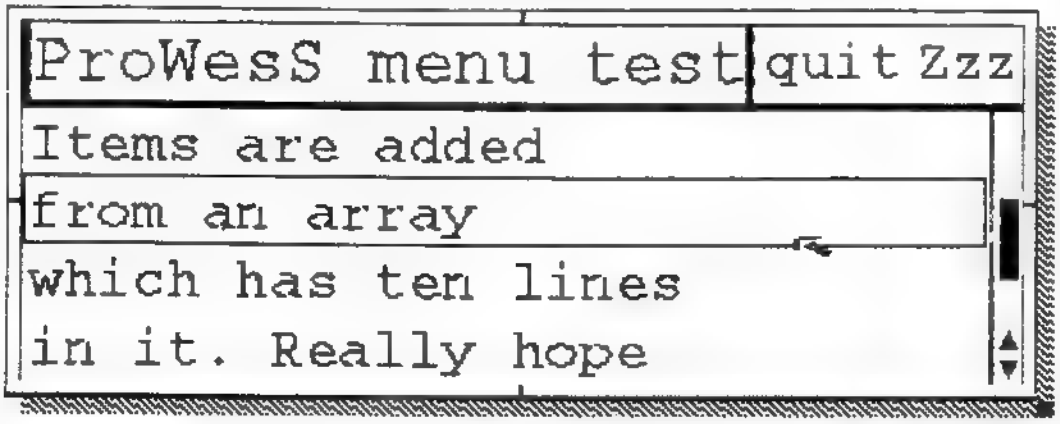


(Figure 1)

Moving the window can be done the traditional pointer environment way (cursor/icon only), or ProWesS can physically redraw the window as you move the cursor. You even have control over how often the window is redrawn during a move. This is useful for adjusting to your system's performance. During a resizing, the traditional method can also be used or better yet, the window frame can be dynamically redrawn as you move the cursor. In this case, the frame is redrawn instead of the full window due to the substantially larger amount of work required to resize a window versus just redrawing it during a simple move. In addition to the everyday improvements, PROGS is also addressing the problems that many users have with installing and using such capabilities. As was learned from the pointer environment, many people did not or could not handle writing boot programs to take advantage of the new features. Therefore PROGS, with the full release version, will be including software to install both ProWesS and any program written to use ProWesS.

Lets Take a Look: A good way to illustrate ProWesS is to look at some screen captures. Figure 1 (much better in color) shows a ProWesS program running along side a program which requires the original window manager, Wman. The two systems are compatible. You also see buttons, which ProWesS also can use. The program shown is a handy little utility that comes with the ProWesS package. It allows the user to dynamically change the configuration of ProWesS and programs using it. It is also very useful for experimenting with different settings, finding just what combination is the most pleasing to you. The new settings become valid as soon as a new window or a part of an existing window affected by the parameter is redrawn.

The line at the bottom of the screen in figure 1 shows the name of a typical parameter and the color values assigned. Notice the parameter name is in plain English and therefore easy to remember. The color is designated in RGB values. The first number stands for the Red value, the second for the Green value, and the third for the Blue value. By combining these three colors, you can create any color you want. A value of 100 0 0 is red, 0 100 0 is green, 0 0 100 is blue, 0 0 0 is black, and 100 100 100 is white. The software is automatically ready for the upcoming higher color/resolution graphics. For the current 4/8 color limits, ProWesS generates different stipple type patterns for the different RGB combinations. You can also observe different font types and sizes being used. These again are set globally in the ProWesS configuration file.

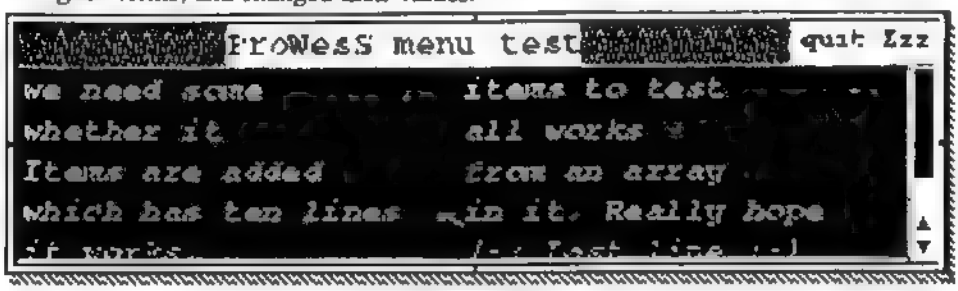


(Figure 2)

ProWesS - (cont'd)

Figure 2 was generated with the default configuration file that was shipped with the package. As you can see, the larger font size makes it easy to read. ProWesS adds objects for all kinds of things, including Titles, Menus, Loose Items (Quit and Zzz), scroll arrows, etc. Each object comes with its own configuration items, which may include font type and size, ink and paper color, border size, etc.

Figure 3 is the same program as shown in Figure 2. However, it looks substantially different. All the changes were made by simply changing the ProWesS configuration file. I chose smaller text to get more information on the screen. Visually, I prefer a dark background and light text. I also decided that I wanted a different font in the menu area. Very simple. I found the items that I wanted to change in the configuration file, easy to do as they are all in simple English terms, and changed their values.



(Figure 3)

Also note that there are now two columns instead of one in the main part of the window. ProWesS automatically configures the windows to fit your configuration choices as best as possible It does this with its own set of rules and can also use hints provided by the software programmer. The resizing was automatic and took no extra code by the program that was running. ProWesS will even supply scrolling if a window becomes too large to display all at once.

What You Get: As I mentioned, if you order ProWesS now, you will get the pre-release version. PROGS will automatically send you the final release when it is available at no extra charge. However for this to happen, you MUST register it (always a good idea).

The pre-release package includes the libraries and system extensions; DLL Manager, system library (Syslib), graphics/fonts handler and drivers (Proforma), window manager (ProWesS and types), scrap extensions from Jochen Merz, and ptr_gen & hot_rext from Tony Tebby. Included utility programs are the ProWesS reader (a hyper-text HTML subset viewer for manuals and help files), dircopy, a calculator, and the ProWesS loader (with associated commands). Four different fonts and their varients (roman, bold, italic, bold-italic) are also included.

The final release will add installation software, extras commands for the ProWesS loader, automatic button installation software, configuration utilities, and some other utilities yet to be determined.

To write programs to use ProWesS, you will need a current copy of C68 (available as freeware from many BBS and most PD suppliers). The programming manuals will not be included but will be available at no charge from many BBS and PD suppliers. They will be readable and printable with the included ProWesS reader.

Users in general will probably also want software which includes a button frame such as QPAC2 so that the sleep functions will handle predictably.

Problems: As with any Alpha release, there were some bugs. And the documentation was limited. A description of changes between the Alpha and pre-release versions shows that most bugs have been fixed, along with other improvements such as speed enhancements. With PROGS track record, the upcoming full release should be very 'clean'. I also expect that the pre-release version will be very usable and not have any major bugs or problems.

At this time, PROGS is not guaranteeing proper functioning with MINERVA due to the MINERVA limitations on the supervisor stack size. I believe that this issue is still being worked on.

ProWesS - (cont'd)

Now for a personal and strictly subjective observation as to font quality. While many users prefer larger fonts for easier viewing (which work great in this package) I personally prefer smaller fonts (at lease until the late night hours). As with any vector based package with limited resolution images, font quality appears to suffer when it gets too small. However, I have faith in ProForma to find a font(s) that holds up well in the smaller sizes. This problem occurred previously with the print drivers under ProForma when using PFList. PROGS came up with a list of different fonts that worked quite well even when reduced to very small sizes.

A word of caution, if you are using an original QL with no acceleration, you might find your computer's horse power a bit too limiting for ProWesS. PROGS recommends a Gold Card QL or better. A hard disk is not necessary but is recommended. An ST with 2MB of memory also works, but may be a bit slow.

Wrap-Up: I have tried to cover a lot of territory in just a few pages. My intention is to briefly introduce you to a new software package that will most likely affect most QL users in the near future.

If you are a programmer, this is a must buy. It incorporates the world of objects into QL programming, which makes window environment programming much easier. Even if you have resisted programming in 'C' before now, it may be time to rethink your choices. This package makes a world of difference. After personally struggling with the QL's windowing environment in QBasic, Assembly, and 'C', I find this package to be a promising and welcome change. More on this in the next issue.

If you are strictly a user, you may choose to not buy this package today. But believe me, I suspect you will be buying it in the near future. If ProWesS is accepted as I think it will be, new programs and rewrites of old programs should start appearing soon. Remember, to use these new 'style' programs, you must own a copy of ProWesS itself.

I have decided to finish the long overdue next version of the BBS reader under ProWesS. I hesitate forcing anyone to buy software to support a program that I develop. However, as with Tony Tebby's ToolKit II, some support software packages are invaluable and become part of the defacto standard.

Also, if you use any of PROGS excellent software packages, you should note that they are planning to update them as time allows to use ProWesS. Just think, color printing from LineDesign (just a hint of possible things to come, as color is possible with the new Proforma).

In the next issue we will start our exploration of programming for ProWesS and discover some of its modern programming constructs. You might want to get a copy of ProWesS before then so that you can follow along. If so, you should also make sure that you get a current copy of C68.

Jim Hunkins E-Mail: jdhunki@ibm.net Compuserve Add: 72567,3624 also on QBOX USA

LINEdesign ('Mirror' Function)

Purley, Surrey, ENGLAND - W. P. J. Baily

I noticed in the article describing the use of LINEdesign to make greeting cards that the author had not found a 'mirroring' or 'reflecting' function within LINEdesign (J D Hunkins, IQLR, Vol 5, Issue 5, January/February 1996, p29).

It is hidden, if that is the right expression, in the EDIT - TRANSFORM OBJECT - SCALE OBJECT routine. In the SCALE quantities a value of -1 for X-scale will give a 'reflection' of the object about a vertical line and similarly a value of -1 for Y-scale will give a 'reflection' about a horizontal line.

(If both X-scale and Y-scale are -1, the result is the same as rotating the object 180 degrees apart from the position of the object on the page).

Quo Vadis

Computer Consultancy/Services
Proprietor, Bruce Nicholls
57 Shaftesbury Road, Romford,
Essex, RM1 2QJ, UK
Tel/Fax: (01708) 755759

QL SOFTWARE

GRAPHICAL

Mini Graphics Printer	£15
Page Designer 3	£40
Scanned Clipart 1&2	£10
Vision Mixer I	£10
Image Processor	£15
Convert-PCX	£10
The Clipart	£12
The Painter	£25
QRactal	£20
ST-Q-QL	£12.50
3D Terrain	£12.50
Quick Mandelbrot 3	£15.00
Image-D	£10
Sidewinder Plus QL	£24.95
Open World	£18

TEXT

Banter	£15
Sidewriter	£15
DeskJet A5	£12
Quick Posters	£10
Text87 Plus4	£79
2488 printer drivers	£29
Typeset94 Deskjet driver for Plus4	£29
Fountext94 + Founted89 for Plus4	£39
Publisher Pack driver for Plus4	£30
(Drivers for use with Linedesign 2)	
Qindex	£20

UTILITIES

Screen Dazzler	£10
Dev Manager	£15
Screen Snatcher	£10
Super Disk Labeller	£10
The Cat	£5
QL-PC Fileserver II	£35
Files 2	£12
TaskMaster	£25
TaskMaster Hard disk	£35
Spellbound	£30
Spellbound S/E	£50
Cueshell	£40
QPAC1	£19.95
QPAC2	£39.95
QTYP2	£29.95
4Matter	£14.95
Locksmithe	£14.95
Toolkit 2 (disk)	£19.95
Floppy Disk Utilities	£18
Disk Mate 5	£37

GAMES/SIMULATION

5 Game Pack	£12.50
Grey Wolf	£12.50
Open Golf	£12.50
Return to Eden	£17.50
Quiz Master 2	£10
The Fugitive	£9.95
Fleet Tactical Command II	£39.95
FTCII Data Print Utility	£9.95
Flightdeck	£15
Spectrum Emulator ZM/128	£28
Spectrum Emulator ZM/HT	£40

PROGRAMMING

Basic Reporter	£10
QLiberator V3.36	£50
QLiberator budget version	£25
QLOAD & QREF	£15
Resident Program Manager	£10
EASYPTR 3 Part 1	£40
EASYPTR 3 Part 2	£20
EASYPTR 3 Part 3	£20
DISA 2	£40
DEA disassembler	£26
MasterBasic	£22
Q-Library Manager	£18

MISCELLANEOUS

Amd-Airplan	£10
Flashback	£25
Flashback S/E	£40
Music Manager	£12

NEWS

**** PROGRAMS NOW AVAILABLE FROM QVD****

Software from Chris Boutal

GENEALOGIST 3	- NOW ONLY £45
GENEALOGIST 2	- ONLY £25
GENEALOGIST BUDGET	- ONLY £10

Software from Geoff Wicks

SOLVIT-PLUS 2	- £15
QL-THESAURUS	- £15
STYLE-CHECK	- £15
QL WORDS PACKAGE	- £25
(Solvit-Plus 2 + QL-Thesaurus)	
QL WRITER'S PACKAGE	- £25
(QL-Thesaurus + Style-Check)	

TERMS/CONDITIONS

Software is supplied on 3.5 DD disks. For software available on microdrive see catalogue. All prices shown are in UK pounds Sterling. Software is sent post free in the UK, overseas add £1.00 per order. Please make payments payable to 'Quo Vadis Design' Payment in UK pounds Sterling currency only. Cheques (drawn on a UK branch of a bank or building society), Postal Orders, International Postal Orders and Eurocheques are all accepted. Goods remain the property of Quo Vadis Design until full payment has been received. Call or write for a more comprehensive catalogue

FWD Computing

(formerly Mechanical Affinity)

P.O. Box 17

Mexico, IN 46958 USA

317-473-8031 Tuesday thru Saturday only, 6 to 9 P.M

FAX 317-472-0783 7 P.M. thru 11 A.M.

Internet E-Mail address: fdavis@walnut.holli.com

Cash, checks, money orders, or COD. Payable to F. Davis.

Add 10% for foreign currency exchange.

C.O. D. Fee will be added to C.O.D. orders.

Postage for North America is included in price.

We do not accept credit cards; $10 charge for Electronic Wire Transfers.

Please allow adequate time for check clearance before shipping.

PRODUCTS FOR QL

Z88 with Z88 to QL Setup - *Next thing to a portable QL. Used, but good Z88, with QL to Z88 File Transfer Programs, 128K extra RAM, Soft Carrying Case, 32K EPROM, and Z88 Source Book. All of this for $199.*

SMSQ/E - *The newest and most extensive operating system ever for the QL. Has Pointer Environment, SBASIC, and Tool Kit 2 built in. Three versions available: QL, QXL, Atari-QL Emulator. Price $150 each.*

QPLANE - *The Powered Back Plane for the QL is in stock. It utilizes a PC Power Supply Unit to help you place your QL motherboard, drive interface, Qubide, etc. inside a PC tower case or full sized desk top case. Add a Super Hermes, Falkenberg Keyboard Interface, or one of our new Di-Ren Keyboard Interfaces plus an IBM style keyboard and it is set to go. Qplane price $52.*

SPECIAL COMBO of QUBIDE and QPLANE - *This includes the Qubide IDE/AT hard drive interface and the Qplane for only $160. Give your QL an up-date and power as a personal computer!*

DI-REN QL KEYBOARD INTERFACES - *This will allow you to use a 101 or 102 key AT keyboard (name brand is recommended) with your QL. This is a very small size board and is easily fitted. It translates most keys to QL format and offers keyboard record/playback facilities. The price is $55.*

AMADEUS QL CONTROLLER - *Designed to link the Sinclair QL to the Amadeus system. This device connects to the QL's ROM port thus enabling high speed communications. Comes with a through port allowing other devices using this to continue to function. The price is $70.*

AMADEUS AMA-SOUND - *Record and play back sounds via your computer. This device employs 12 bit sampling and gives the high quality audio of the ADPCM algorithim. Recorded files may be stored, edited and replayed. Includes all hardware and software. Sample data is in 4 bit packages. All data can be transferred between different types of computers. 3 bit sampling may also be employed. The price for this great innovation is $84.*

QL KEYBOARD MEMBRANES - *Replacement membranes for $18.*

QL POWER SUPPLIES - *Get a backup or replacement for $16 while they are still available. These are 110 volt. The supply is limited.*

The New England Sinclair QL Users Group (NESQLUG)

Hosts - The Fourth North American QL Show

Saturday, May 18th, 1996 From 9 AM until 5 PM

In Bedford, Massachusetts, just 15 miles NW of Boston. The QL Show will be held at the: Bedford Ramada Inn 340 Great Road - Bedford, Massachusetts 01730 -Tel: In US (800) 228-2828 or (617) 275-6700. Fax: (617) 275-3011

A block of 25 rooms has been reserved for Friday and Saturday the 17th & 18th of May, 1996, for those attending the show, at a reduced price of just $49 per night - 1 or 2 persons, and $10 more for an additional person. This price also includes an all-you-can-eat, American buffet breakfast. Please mention the 'QL Show' in order to get this special price. Additional nights are $59 per night for 1 or 2 people. Rooms must be reserved not later than April 27th, three weeks prior to the meeting. If you reserve a room at the Ramada by April 27th, you may pay the $5 entry fee at the show. If you are not planning to stay at the Ramada, please send a check for $5, by May 7th, made out to:

Gary Norton, 43 Richardson St., Billerica, MA 01821 USA.

Late entry fee at the show is $7. Notify Gary also, if you would like a packet with maps and tourist information. E-mail: norton@prevline.health.org, tel: 508 667-2048, or mail as above.

THE AGENDA: Friday, 17 May (Optional) - Meet in the Ramada Lounge at 7PM, and share rides to the Willow Pond Restaurant in Concord, an informal pub that offers meals under $5 and twin lobster dinners for $14.95.

Saturday, 18 May - 8:00 AM Doors to meeting room on 3rd floor of Ramada opened to vendors. 9:00 - Noon General meeting - Coffee and tea will be provided. Noon - 1:00 Meeting room closed for lunch break. (*) 1:00 - 4:30 General meeting. 4:30 - 5:00 Vendors remove equipment from room. Numerous valuable QL hardware/software doorprizes will given away all day !

(*) Note: Only sandwiches are available for Lunch at the Ramada Inn, but there is a variety of eating places within 500 yards of the meeting location.

Saturday night QL Banquet : Cost is $19.95. IMPORTANT: Contact Gary Norton to reserve your seat not later than May 7th. If you reserve a room at the Ramada, you may defer payment until you arrive. Otherwise please send a check made out to Gary Norton. NOTE: Banquet is limited to first 60 QLrs.

6:00 - 6:30 Reception - 2nd floor Banquet Room, cash bar (open all evening) 6:30 - 7:45 QL Banquet

BANQUET MENU:

Fruit cup, Garden salad.
Choice of: Broiled Boston Scrod (a tasty white meat fish) or Roasted Chicken with Supream Sauce.
Roasted Potatoes, Green Beans Almondine,
New England Shortcake with Strawberries,

7:45 - 7:50 Awards Presentation
7:50 - 8:00: A QL Quorum - a panal of knowledgeable volunteers will answer questions from the floor.

The following vendors and QL notables have indicated they are planning to come: Stuart Honeyball - Miracle Systems, Jochen Merz - Jochen Merz Software, Frank & Carol Davis - F W D Computing and UPDATE Magazine, Bill Cable - Wood and Wind Computing, Roy Wood - Q BRANCH, Tim Swenson - QL Hacker's Journal, Tony Firshman - TF Services and Robin Barker -House of Di-Ren.

COME JOIN US for a QL GOOD TIME !!

OF MICE AND MENUS

Portslade, Sussex, ENGLAND - Roy Wood

Ah mice. Little furry creatures running around and squeaking and women standing on chairs in early sixties sitcoms or little plastic objects that either aggravate or enhance the lot of computer users. Before you turn the page and read something else let me say that this article is not an attempt to get people to use mice when they do not want to - quite the reverse in fact - I would like to explain to those of you that are not mouse users why you do not have to reject the Pointer Environment altogether.

OK so it is a Bad Name: The pointer environment has an air, for many people at least, of being the vehicle for the mouse culture. 'You just have to have one' - but that is not the case. Most programs written for the Pointer Environment use the full range of function keys and combinations of keystrokes to give the commands that they need to operate so there is no need to use a mouse at all. I know this from personal experience because, when I first got my portable Epson computer and installed a mouse driver I managed to choose one of the few mouse drivers that will not work with the QXL (Digitus). The advantages that you get from using the pointer environment lie in it's handling of windows and the ease with which the system multitasks. The use of a pointer to react with objects on a screen can, in many cases, make the handling of the program more transparent than a string of keypresses that have to be remembered. Some people, though, can store all these keypresses up and that leads us to the crux of the argument.

Everyone uses their computer differently. Whatever you originally bought the computer for, you have by now, found two hundred other uses for it. We all came to the point of being sufficiently interested in the computer itself that we are reading this magazine from totally different directions. It is, therefore, natural that we should have different approaches to the way in which we work. Apart from these considerations there is also the important factor of what task are we actually trying to achieve. Some things suggest themselves immediately as being either mouse or non mouse oriented. Word processing lends itself to the use of the keyboard and anything which takes your hands away from it will slow you down but then a large part of word processing is actually looking at the text and reformatting it to get the final presentation. At that point you are usually navigating the text with the cursor keys and that is something a mouse can do much easier, so there is no easy answer.

I use two main text processing devices, Text87 which is not pointer driven and is used mostly for writing articles, letters etc. and QD 8 which is pointer driven and is used for writing programs and other pieces of text that require that there be no control codes in them. Each of them has it's own function and it is interesting to compare the way in which the structure of the program forces the user to work differently. In Text87 everything has to be done from the keyboard. To load a file from the initial screen you have to press 'L' and then type in a file name followed by pressing 'ENTER'. The same kind of process is repeated whatever the action you wanted to perform although some of these can be put into 'macros'. This, in itself, is not too much of a chore but the point where it all falls down for me is when I want to mark a particular block of text and copy it or delete it. QD 8, on the other hand, allows me to leave the keyboard and use the mouse to position the beginning and end of the block and then perform the action on it. These two different approaches to the same function need not be mutually exclusive. QD is mouse driven but you can use the cursor to travel to the start, press F7 to indicate the beginning and travel to the end and use F8 to indicate that. You then have to press F4 for the block commands and D for delete. QD 8 also provides shortcut macros to do these functions so, once the block is marked, CONTROL/K would also have deleted it.

I employ these two differing methods all the time when using these programs and I would be reluctant to say which was the faster of the two. In QD I lean towards the use of the mouse for all of the text navigation and for most of the file operations but I use the cursor and the function keys equally often. I suppose that is where the well written pointer program has a distict advantage over the non pointer one - it works in a variety of ways. A lot of the pointer programs available at the moment use combinations of 'CONTROL' and a character key to provide shortcuts to the program's functions and the 'F' keys (F1, F2. F3 etc.) to initiate the menus. These keypresses can be memorised just as easily as the keypresses for other programs. I have used Jochen Merz's QMenu extensions to write a short program that can be popped up over T87 to give pointer driven menus for inserting strings into the text and for file selection so I have, in effect, added mouse control to part of a non pointer program. Even Qmenu, which is an 'extension thing' which allows you to write pointer driven menus, also allows you to add a key letter to the functions to eliminate the use of a mouse.

IT IS A MEMORY THING: Most programs need a lot of commands to operate. We have come a long way from the early programs that have just a few commands and today's programs have long lists of commands and keywords. You may be one of those people who will always effortlessly remember sequences of letters and be able to recall all of the functions that the program is capable of or you may be grateful when a little menu pops up to give you a hint. A member of our user group, John Roberts, can type away at speed in 'The Editor' calling up all manner of sorting and block manipulation functions without batting an eyelid. I find I get so tied up in whatever it is I am doing that the key combinations tend to get shuffled off to the side and that is when the menus come in handy.

People who automatically dismiss the Pointer Environment are missing out on a lot of programs which are vast improvements over the non-pointer ones, most of which were written a few years ago. One thing most pointer programs are capable of is an expansion of the program's windows. This will gain an increasing importance when the new graphics card arrives. Until a few years ago most QL programs were written for a standard 512 x 256 screen but the emergence of the QVMEand QXL QL emulators for the Atari and PC as well as the new operating system, SMSQ/E, have meant that a lot of programmers have now used the Pointer Environment's window handling facilities to give the user the opportunity to use larger screen sizes and higher definition. This, in itself, will make using the pointer programs a more attractive proposition than the non-pointer ones.

The computer world is becoming increasingly pointer oriented and fewer and fewer major new programs are written without a pointer interface. There are programs that use standard menus and require the pointer interface loaded but do not use a mouse. Some of these come from the Italian software house Ergon. These are very well constructed pieces of code that perform a number of tasks and, although they use a system of pull down menus the control of the program is assigned totally to the keyboard. The pointer environment is used mainly to control the window handling .

A GRAPHIC EXAMPLE: There is, of course, one area where the mouse reigns supreme and that is graphics. If you have ever tried to use your computer to draw, using any one of a number of graphics programs that are available, you will have realised that the cursor keys provide at best a clumsy interface to the screen. When I got my first QL (and that was long before there was a mouse available) I used QL Paint and I found drawing complex objects very hard to do. I later bought what I think was the first mouse system for the QL: ICE. This was a plug-in ROM which had a mouse attached and provided mouse control of the programs that were written specifically for it. This made drawing easier, and their programs, 'ArtICE' and 'Drawing OFF-ICE' were very simple to use.

No one would deny that graphics is one area where the use of a mouse is almost mandatory; another place where the rodent comes in handy is in a spreadsheet. A lot of spreadsheet use is keyboard based but the entry of data and formulae is only part of the use of the sheet. Spreadsheets are not just a list of items and figures that are automatically calculated with regard to the formulae entered. The most useful features of the spreadsheet lie in the way in which you can change one area and see the result on the whole sheet. This can be done very easily by a mouse. There are two spreadsheet programs available for the QL. The one we are all familiar with is Abacus the Psion program that was provided with the original QL package. The other is QSpread, a pointer program written by Oliver Fink in Germany. The current version of QSpread (v1.29) does not have some of Abacus' more unusual features (such as order rows/columns) but is mouse controllable, so block mark/move/copy functions become very quick and efficient. The program also has scroll bars along the side and bottom which allow the user to move quickly from one part of the sheet to the other. Both of these examples are essentially graphic in their approach because they require the user to move around the screen rapidly , but that same requirement is evident in many other programs too.

"Put Your Money Where Your Mouse Is": When it comes down to choosing a mouse system to buy for the QL the choice is quite limited.In choosing the system you should also take into account the different ways in which the mouse systems operate. For a while, the pages of QL World and Quanta echoed with the cries of people who could not 'get this or that mouse to work' or who had written their own mouse drivers but truly commercial mouse systems were thin on the ground.

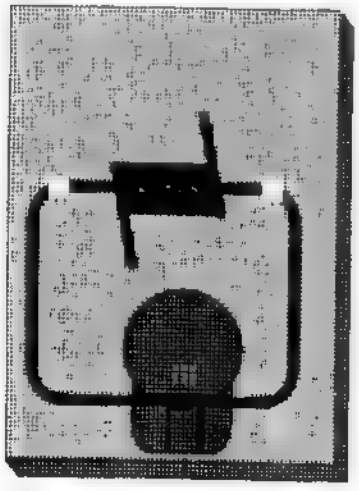
The ICE mouse became obsolete a long time ago because the only programs that supported that particular mouse

were those produced by Eidersoft. The software house did, however, produce a program called Icicle which allowed the user to modify other programs to have pop-up menus which were then activated by the mouse and the appropriate actions performed. Surprisingly enough no-one has ever taken this concept on board for any of the other subsequent mice systems. Icicle could adapt a program like Quill or Archive to have a mouse driven menu system and the instructions chosen by the mouse could then be passed directly to the host program. I have attempted to do similar things with QMenu with varying degrees of success although I now no longer use the Psion programs and have, almost entirely, gone over to the pointer driven replacements.

One mouse of which there is hardly ever a bad word heard is the Qimi from Quanta. This system plugs directly into the QL's motherboard and comes complete with its own 2 button mouse - no software to load. I have never used one of these myself but everyone I know who has is entirely satisfied. The only drawback to this system is, I feel, that the mouse is a two button version which allows you to have the left button as 'HIT' the right button as 'DO' and the two together as the altkey to pick the button frame (essential for QPAC 2 users).

Albin Hessler took things a stage further with his Sermouse which is a software driven serial mouse that plugs into one of the two serial ports on the QL. The software for this can be configured to give any number of different actions from any of the three buttons on a standard mouse as well as combinations of those buttons. You can, therefore, program the centre and left buttons to give you CONTROL/F2, for instance to re-read the directory in the QPAC 2 files menu, or CONTROL /F3 to move the window. There is even a section in the configuration blocks that will allow you to configure the button's actions when the pointer environment is not present thus allowing the mouse to become a cursor. There are, unfortunately, drawbacks with this system too, since there are a few mice out there which steadfastly refuse to behave as they should. The QL's serial ports are also a problem with regard to this software. They are, in fact, one serial port or siamese ports (joined at the hip so to speak)so any attempt to read from one whilst writing to the other is a source of potential disaster. T.F. Services Hermes chip went a long way towards fixing these little problems but using communications programs like QTPI (pointer driven and much easier to use with a mouse) produced utter garbage on the screen until the mouse driveris turned off.

For a long time this was no problem for me since I did not have a modem but I have, just recently taken my first steps towards acquiring one of these beasts and it became obvious I needed a replacement for my long serving SERmous. T.F. Services again came to the rescue and now my main QL sports the new Super Hermes Chip. It was plagued with problems from the start and what must have seemed like a complex but reasonable task last year had become a massive migraine just before Christmas 1995. A bug on the actual chip caused the launch date to slip back a couple of months. Then keyboard lock-ups and a whole host of other relatively minor glitches resulted in a lot more work and the re-writing of whole sections of code. Fortunately T.F. are patient and persevering people and the chip that now lies at the heart of my QL does all that I want it to and more besides. I now have two extra serial ports, one high speed and one dedicated to my mouse itself in addition to the original two provided by Sinclair and there is a keyboard interface to boot.(which, during some of the keyboard lock-ups I had at the start, is what I felt like doing)



One last nibble at the Cheese: So there you are. As I said at the beginning, I am not trying to sell you on the idea of a mouse if you do not want one. I would not be without mine but that is just the way I work. What I do say is, try a few pointer programs and then tell me they are not as easy to use as the non-pointer ones - I am willing to bet that you cannot. Sooner or later you may find you might want to try using a mouse and after a bit of hand/eye co-o rdination practise you could be hooked - or is it mousetrapped ?

Converting File Formats

Aberdeen, SCOTLAND - Norman Dunbar

As the author of a number of QL programs, I recognise the need for a clear, concise and well laid out user manual. To produce my own manuals I (now) use a desk top publishing package on my PC which gives a very good quality manual indeed. There is one fly in the ointment, however, the screen dumps need to be converted to something that the DTP package can recognise.

To achieve this, I wrote a program using C68 to convert a single QL screen dump file into a Windows 'bitmap' file. This program has been used to good effect in my little booklet 'The Idiot's Guide To The Pointer Environment' which was printed in IQLR a few issues back. All of the screen dumps in the article and the booklet were converted using SCR2BMP.

I usually keep a single floppy disc with all the required screen dumps on it for each manual that I produce, I therefore wrote the program so that it would accept a list of filenames to be converted and later on I added a mode number and an output device name so that I could convert all my screen dumps in one go. For example :-

```
EX Scr2Bmp ; 'flp1_*_scr 4 ram1_'
```

will take every file on 'flp1_' which has a name ending with the characters '_SCR', and convert them from mode 4 to a file on 'ram1_'. The output files are the same name as the input files except that the suffix '_BMP' is added to the name.

The program accepts wildcards in the filename(s) that you supply. As in the above example, and asterisk (*) replaces any or no characters so if there was a file simply called flp1_scr and another called flp1_xyz_abc_scr then both would be converted. A question mark (?) replaces only 1 character.

If the command had been

```
EX Scr2Bmp ; 'flp1_Manual_?_scr 4 ram1_'
```

then only those files which have a filename of the form Manual_1_scr, Manual_9_scr etc will be converted from flp1_ to ram1_.

As it stands, Scr2Bmp will not work properly unless the screen on your machine has its base address at 131072 (decimal). This rules out the QXL unless it has been configured for QL mode. Having said that, it does work in VGA mode but you don't see the screen when it loads and who knows what is being overwritten at that address ! The mode number must be 4 or 8 at present.

Please note that the SCR2BMP program does not write the converted files to an MS-DOS disc, this still has to be done as an additional step prior to loading them onto my PC's hard drive for eventual inclusion in the finished manual.

Before we consider the actual program, we need to know about how the memory used by the QL for its screen display is laid out. We also need to know how the PC stores its bitmap files.

QL SCREEN FORMAT: The QL has 2 modes for its screen, mode 8 and mode 4. I know that there are other modes out there on various emulators and in SMS etc, but I am dealing with the simplest case, the standard QL. After all, I wrote the program for my own use and I don't have any other mode available !

In mode 4 there can be only 4 colours with a screen size of 512 across by 256 down. In mode 8 there can be 8 colours with a screen size of 256 by 256 pixels across and down. In both modes, the origin is in the top left of the screen and this is pixel (0,0).

Checking with QDOS technical information we see that in mode 4 each set of 8 pixels is defined by the data held in 2 adjacent words. The first byte holds the green data while the second holds the red data. The first or green byte has an address which is even and the second or red byte has an odd address which is always 1 higher than the green byte.

The format of the mode 4 colour word is this :-

```
-------------------------------------------------------------
G0 G1 G2 G3 G4 G5 G7 G7 ¦ R0 R1 R2 R3 R4 R5 R6 R7
-------------------------------------------------------------
```

G0 is the green bit for pixel 0, R0 is the red bit for pixel 0, G1 is green for pixel 1 and R1 is the red for pixel 1. Pixel 0 is the one closest to the left side of the screen, pixel 1 is next to it and so on. Each word, in mode 4 controls 8 adjacent pixels on the screen.

Each combination of a green bit and a red bit gives a binary range of 00 to 11 or four different values. These values correspond to the four colours allowed in mode 4. The four colour values are :-

Binary	Decimal	Colour
00	0	BLACK
01	1	RED
10	2	GREEN
11	3	WHITE

So, putting on our binary thinking hats for a moment, we see that a pixel's colour is defined by the 2 bit binary number which is made up of the green bit followed by the red bit. All the combinations allowed are shown in the table above.

Using the fact that we have only 2 bits per pixel, shows how a single word can be used to define the colours for 8 pixels on the screen. This memory layout can lead to all sorts of head numbing problems when you try to figure out all the shifts and masks that need to be done to get the 2 bits into a register in the correct order, but it can be done.

(You need to mask out the bits you don't want from each byte of the word, shift the red bit into bit zero of the byte, shift the green bit into bit 1 of the byte then OR the two together to get the 2 bit value - easy to say, try to work it out before looking at the program.)

Mode 8 is a different kettle of fish altogether. we have 8 colours and flashing to think about. Once again the data is combined in two adjacent bytes of memory but the layout is quite different. The format of the mode 8 colour word is this :-

```
-------------------------------------------------------------
G1 F0 G1 F1 G2 F2 G3 F3 ¦ R0 B0 R1 B1 R2 B2 R3 B3
-------------------------------------------------------------
```

Each pixel has a green bit, a flash bit, a red bit and a blue bit. The flash bit is not used to define the final colour of the pixel so we are left with 3 bits giving a total of 8 different values. On the PC the flashing is ignored so the SCR2BMP program simply masks them out and ignores them. The eight colour values are :-

Binary	Decimal	Colour
000	0	BLACK
001	1	BLUE
010	2	RED
011	3	MAGENTA
100	4	GREEN
101	5	CYAN
110	6	YELLOW
111	7	WHITE

So, putting our binary thinking hats back on, we see that a pixel's colour is defined by the 3 bit binary number which is made up of the green bit followed by the red bit followed by the blue bit. All the combinations allowed are shown in the table above. Once again, think how we will mask and shift the various bits from the 2 bytes to get a 3 bit value.

Figured it out yet ? Mask out all the bits in the green and red words that we don't need and don't forget that we need 2 bits in the red word this time. Now shift the red and blue bits right until the blue bit is in bit 0 and the red bit should be in bit 1. Shift the green bit into bit 2 of its byte and OR the two bytes together to get a single 3 bit value. Easy ?

Using the fact that we need 3 bits per pixel plus a flash bit, a single word can therefore define the colours for only 4 pixels on the screen in mode 8.

That is the end of our brief tour of the QL's display memory map. The screen dump files that we will be working with are all assumed to be a 'standard' 32 Kbytes long and saved using something like SBYTES filename, 131072, 32 * 1024 which will work on a 'normal' QL or a QXL in QL mode.

The files are a simple binary image of the QL's screen memory and are identical to it in every respect. The PC is not so simple.

PC BITMAP FILE FORMAT: The file format for a PC bitmap is different to how it is stored in the actual memory of the PC. However, having said that, we don't care how it is held in memory as we are only concerned with how it should look on a disc.

Additionally, you must remember that a PC is daft ! When a word or long word of data is stored, the bytes are stored in reverse order. A word is stored as low byte then high byte. A long word is low word then high word with each word reversed as above. The QL stores words and long words in the 'correct' order with the most significant byte first. For example :-

On the QL, the word $1234 is stored as two bytes, $12 followed by $34 with $12 being at the even address (words and long words must be on even addresses) and the $34 bytes being stored at the address 1 higher in memeory. This is exactly as you would write it down with the most significant byte written first on the paper and as an analogy, first in memory too.

The PC stored the word $1234 as $3412, note how the $34 byte is stored first followed by $12. As this goes it is a bit sad, however, long words are worse. The QL is quite happy for $12345678 to be stored as 4 bytes in a row, $12, $34, $56 and finally $78. This is equivalent to storing the two words $1234 and $5678 and is so simple.

Guess what the PC does ? First it stores the low word in back to front order as $7856 followed by $3412 giving a long word stored as the 4 bytes $78, $56, $34 and finally $12. The question has to be WHY ? (I remember the Z80 doing something similar from by Sinclair Spectrum and ZX81 days.) Anyway, enough of this. A PC bitmap file is made up of 3 different parts, as follows :-

A BITMAP FILE HEADER
A BITMAP INFO
The BITMAP DATA

The BIT MAP FILE HEADER is a simple header that tells the reading program that the file is a bitmap and how long it is (supposed) to be. The structure is :

WORD - must be 'BM' in upper case
LONG - the size of whole file in bytes (some manuals define this as the size in long words, they are wrong !

WORD - must be zero
WORD - must be zero
LONG - the offset in bytes from start of file to the bitmap data.

The BITMAP INFO part of the file is made up of 2 separate parts :-

A BIT MAP INFO HEADER
The RGB QUAD values

The BIT MAP INFO HEADER is a header that tells the reading program all about the actual bitmap that the file contains. It has the following structure :-

LONG	- the size, in bytes, of the BIT MAP INFO HEADER must be 40.
LONG	- the width of the bitmap, in pixels, for us it is 512.
LONG	- the height of the bitmap in pixels, again for us it is 512.
WORD	- the number of colour planes. Must be 1.
WORD	- the number of bits per pixel. Must be 1, 4, 8 or 24. We use 4.
LONG	- the compression used. we set it to 0.
LONG	- the size, in bytes, of the actual image data.
LONG	- the number of pixels per metre wide, best set to zero.
LONG	- the number of pixels per metre high, best set to zero.
LONG	- the number of colours used in the palette, Set to zero.
LONG	- the number of those colours which are important. Set to 16.

The Compression field tells Windows that the device driver (ie the software to work the monitor on the PC) should decompress the data as it is sent. Many drivers cannot do this due to bugs, so most bitmap files have no compression on them. We will adopt the 'no compression' method for our conversions.

The bits per pixel field, defines the number of bits used in the actual data to define the colour for each pixel. This also gives the maximum number of colours as 2 ^ Bits Per Pixel. We use 4 bits to get 16 colours which is the smallest value that allows us to use 8 colours maximum.

The number of pixels per metre fields are usually set to zero. They provide scaling information for various Windows display devices, plotters etc. This program does not use them.

Following on from the BITMAP INFO HEADER are the RGB QUADs, one for each colour in the file. In our version, there are 16 RGB QUADS to define the 16 colours. As a QL only has 8 colours, we can use the 16 colour format to save space in the output file. The format of an RGB QUAD is :-

BYTE - Intensity of Blue, 0 - 255.
BYTE - Intensity of Green, 0 - 255.
BYTE - Intensity of Red, 0 - 255.
BYTE - Reserved, must be zero.

Finally, there is the BITMAP DATA. This data defines the colour of each pixel in the bitmap and uses 'bits per pixel' bits to define each pixel. The colour is not hard coded into the appropriate number of bits, the index into the RGB QUAD table is encoded into the bits.

In our file there are 4 bits to each pixel, so each set of 4 bits is simply the index into the colour table, for the appropriate colour of this pixel. There are 16 colours numbering from 0 to 15 and each colour has been defined in the preceeding RGBQUAD table. Note that the numbers are not the same as the QL colour numbers, but :-

QL	0 = BLACK	= 0 on PC
	1 = BLUE	= 12
	2 = RED	= 9
	3 = MAGENTA	= 13
	4 = GREEN	= 10
	5 = CYAN	= 14
	6 = YELLOW	= 11
	7 = WHITE	= 15

so a certain amount of jiggery-pokery is required to convert our QL colour code into the PC's colour code. The 16 colours in our bitmap file's palette are :- black, dark red, dark green, dark yellow, dark blue, dark magenta, dark cyan, light grey, dark grey, red, green, yellow, blue, magenta, cyan and white numbering from 0 to 15.

In addition, the QL's (0,0) pixel is at the top left, the PC has its (0,0) pixel at the bottom left, just to make things interesting. I discovered this when every file I converted loaded into Windows upside down !

SCR2BMP - The program: I realise that many people cannot code in C and I also realise that many people can code in C far better than I will ever be able to. I write programs that can be maintained and I like to add plenty of comments as I go along, however, to save space in the magazine, most comments have been edited out. The running commentary should keep you informed.

The following code is imported directly from the program and does work. First we start with the comments defining the program and what it does and the history of any amendment made to it. This is always useful in a program especially if you have to come back and make changes to it at a later date.

```
/*=============================================================================*
 * A useful (?) program to convert a QL screen, not yet QXL compatible,        *
 * from address $20000 (131072) to a PC Windows 3.1 Bitmap file (BMP).         *
 *-----------------------------------------------------------*                 *
 * Copyright Norman Dunbar June 1994 onwards.                                  *
 *-----------------------------------------------------------*                 *
 * BUGS OUTSTANDING.                                                           *
 *                                                                             *
 * None (!)                                                                    *
 *-----------------------------------------------------------*                 *
 * HISTORY.                                                                    *
 *                                                                             *
 * 1. First attempt seemed to work ok but colours all wrong on the PC.         *
 * 2. Second attempt, colours ok now, it was a case of not writing to          *
 *    the disc file properly.                                  v1.01           *
 * 3. 23/06/94 - added mode 8 handling.                        v1.02           *
 * 4. 24/06/94 - Doubled pixel width in mode 8 to keep proportions.  v1.03     *
 * 5. 22/08/94 - Added command line parameters. Output adds _BMP.    v1.04     *
 *=============================================================================*/
```

Having got the documentation out of the way, we have some files that must be included for some of the standard C functions used in the program. Novice programmers may not know that before you can use some C functions or procedures, you must tell the compiler about them. If you don't, then all sorts of errors can occur. These files tell the compiler about some 'built in' functions that the program uses. Any that the program defines itself have to be notified to the compiler as well. That will be done later.

```
#include <stdio_h>
#include <qdos_h>
#include <string_h>
#include <stdlib_h>
```

After the includes we have a number of bits specific to C68. These allow the size of the finished executable program to be reduced by not including code to perform certain tasks that the program does not need. In this case, we don't allow channel redirection or channels to be passed as parameters etc.

```
long (*_cmdchannels)() = NULL;          /* No channels redirected */
long (*_stackchannels)() = NULL;        /* Or on stack either     */
int  (*_conwrite)() = NULL;             /* Or output translation  */
void (*_cmdwildcard)() = cmdexpand;     /* Allow wildcard params  */
```

This program, when compiled, appears in a little window with a title bar and its name nicely centred. The following lines set this up and define the 3 fields to be displayed in the title bar. All of this is C68 specific.

```
void consetup_title();                          /* Fancy title window */
void (*_consetup)() = consetup_title;           /* Link it in       */

char _prog_name[] = "Scr2bmp";                  /* Program name     */
char _version[] = "Version 1.04";               /* Version number   */
char _copyright[] = "© Norman Dunbar, 1994";    /* Author & copright */
```

Now we define the functions and procedures used in this program.

```
void write_BMfile_header(FILE *fp);             /* Write BITMAPFILEHEADER */
void write_BMInfo(FILE *fp);                    /* Write BITMAPIFO      */
void write_RGBQuad(FILE *fp);                   /* Write pallette info  */
void load_screen(FILE *fp);                     /* Load QL screen file  */
void convert_screen(FILE *fp, int mode);        /* Convert it to _BMP   */
void set_mode(int mode);                        /* Check & set QL mode  */
```

A C program always begins at a function called 'main' and this one is no exception. The first thing we must do is declare all our local variables for the function. This is very similar to SuperBasic where you must always define LOCals before any other code in a procedure or function.

```
/*=============================================================================MAIN==*/
main(int argc, char *argv[])
{

  char bmp_file[128];                   /* Buffer for QL screen filename  */
  char scr_file[128];                   /* Buffer for PC BMP filename     */
  int  mode;                            /* Holds mode 4 or 8 accordingly  */
  FILE *scr = NULL;                     /* File pointer for QL screen file */
  FILE *bmp = NULL;                     /* File pointer for PC BMP file   */
  int  file;                            /* Loop counter for each parameter */
```

Having declared all the locals required by the 'main' function, we can check to see if there were at least 3 parameters supplied to the program. The first parameter to main is the count of the parameters supplied and the next is an array of pointers to each of the parameters. If there is not enough parameters, a message is displayed on the screen advising the user of the error of his/her ways and the program terminates.

```
  if (argc < 4) {
     printf("USAGE : ex SCR2BMP;'scr_files mode bmp_device'\n\n");
     printf("Where :\n\n");
     printf("scr_files                  = All filenames to be converted to bmp files\n");
     printf("          EG. flp1_*_SCR = all '_scr' files on 'flp1_'\n\n");
     printf("mode                       = Screen mode of ALL the supplied filenames\n");
     printf("          EG. 4 or 8 for mode 4 or mode 8\n\n");
     printf("bmp_device                 = Device where all converted files are written\n");
     printf("          EG. ram1_        = write converted files to 'ram1_'\n\n");
     exit(0);
  }
```

Our next job is to set the mode of the program to be the same as that of the files to be converted. This is required as we intend to load each screen into the QL's display memory as a form of feedback to the user that the program is working fine.

Converting File Formats - (cont'd)

First we extract the mode parameter which is always the second last one, convert it from a string into a number and make sure that it is either 4 or 8. If all is ok, we call one of our own functions which is called 'set_mode' to do the necessary work required in setting the mode.

```
mode = atoi(argv[argc -2]);
if ((mode != 4) && (mode != 8)) {
   printf("Incorrect mode supplied, must be 4 or 8 only\n");
   exit (0);
}

set_mode(mode);
```

Having set the screen mode, we now enter a loop to process each and every file in the list of filenames that are to be converted. The first filename is parameter 1 supplied to the program and the last is 3 less that the total number of parameter the program was given. We process each file in turn. (A C program gets an extra parameter supplied to it. This is always the very first one and is always the program name. It is always found at argv[0].)

First we copy the file name into a buffer called 'scr_file' and then attempt to open the file for reading. If the open fails a message is displayed and the file is totally ignored so we jump back to the start of the loop to process the next file.

```
for (file = 1; file < argc -2; file++) {
   strcpy(scr_file, argv[file]);

   scr = fopen(scr_file, "r");
   if (scr == NULL) {
      sd_clear(fgetchid(stdout), -1);
      printf("%s can't be opened.\n", scr_file);
      continue;
   }
```

Assuming that the file has opened correctly, we create an output file name in the 'bmp_file' buffer. This name is simply the input filename with its first 5 characters removed added on to the end of the output device as supplied in the final parameter to the program. Removing the first 5 characters from the input file should cause the name of the input device to be replaced with that of the output device. This may cause problems if the input device is a network one.

Having made up an output filename, we add the extension '_BMP' to the end and attempt to open it for binary writing. If this fails the program displays a message, closes the input file and jumps to the top of the loop to get the next file processed.

```
   strcpy(bmp_file, argv[argc-1]);
   strncat(bmp_file, scr_file +5, strlen(scr_file) -5);
   strcat(bmp_file, "_bmp");

   bmp = fopen(bmp_file, "wb");
   if (bmp == NULL) {
      sd_clear(fgetchid(stdout), -1);
      printf("%s can't be opened.\n", bmp_file);
      fclose(scr);
      continue;
   }
```

Once the output file has been opened, it is a simple task to set up and write out the various headers for the bitmap file. A call to 3 of our own functions writes out the BIT MAP FILE HEADER, the BITMAP INFO and the colour palette in the form of an array of RGB QUADS.

```
    write_BMfile_header(bmp);
    write_BMInfo(bmp);
    write_RGBQuad(bmp);
```

Now the difficult part begins. Firstly the program loads the QL's screen file into the standard memory address for a normal QL. If you have an emulator or a QXL in non QL mode then this may not work. Having loaded the screen, convert it to a bitmap file then finally close both the input and output files. We are now at the end of the loop so jump back to the top to process the next file.

```
    load_screen(scr);
    convert_screen(bmp, mode);

    fclose(bmp);
    fclose(scr);

  } /* end for */
```

If there are no more files to be converted, clear the rubbish off the screen and exit from the program with no errors. This is the end of the 'main' function.

```
  sd_clear(fgetchid(stdout), -1);
  exit(0);
}
```

Before setting the mode of the screen we need to call a C libray function that can read the current mode. If this is not the same as the mode that we require it will be changed. If it is ok, then no change will be made.

```
/*=============================================================================SET_MODE==*/
void set_mode(int mode)
{
  short s_mode,   /* Current mode */
        d_type;   /* Display type */

  /*-----------------------------*
   * Read the current mode first. *
   *-----------------------------*/

  d_type = -1;
  s_mode = -1;

  mt_dmode(&s_mode, &d_type);

  /*-------------------------------------*
   * Is the current mode the same as we need ? *
   * if not, set it.                     *
   *-------------------------------------*/

  if (!s_mode) s_mode = 4;   /* Zero = mode 4 */

  if (s_mode != (short)mode) {
     s_mode = mode;
     d_type = -1;
     mt_dmode(&s_mode, &d_type);
  }
}
```

The function to load a QL screen into memory is quite simple. It simply reads 256 scan lines from the input file and stores them in the normal QL's display memory. Each scan line is 128 bytes wide.

```
/*=================================================================LOAD_SCREEN==*/
void load_screen(FILE *fp)
{
  /*-------------------------------*
   * Simply read 256 scan lines of 128 bytes *
   * into the QL display memory at 131072.  *
   *-------------------------------*/

  fread((char *)131072, 256, 128, fp);
}
```

The following function writes a BIT MAP FILE HEADER to the output file. This must be the first thing in the file and contains data about the file. This data tells Windows that the file contains a bitmap, how long the file should be and how far into the file the actual bitmap data begins.

For SCR2BMP these values are fixed. The file is 65,654 bytes long (10076 hex which in MS-DOS format is the 4 bytes 76, 0, 1, 0 in hex) The bitmap data always begins 76 bytes in from the start of the file.

This function sets up a bit map file header and writes it to the output file.

```
/*=============================================================WRITE_BMFILE_HEADER==*
/
void write_BMfile_header(FILE *fp)
{
  int x;
  static char bmh[] = {  'B', 'M',          /* Flag for Bit Map            */
                         118, 0, 1, 0,      /* File size in bytes (not DWORDS) */
                         0, 0, 0, 0,        /* Reserved 1 & reserved 2     */
                         118, 0, 0, 0       /* Offset (bytes) to the bitmap  */
                       };

  for (x = 0; x < 14; x++) {
     putc(bmh[x], fp);
  }
}
```

Having stuffed a BITMAP FILE HEADER into the file, we now need a BITMAP INFO structure to be written in its 2 constituent parts, the header and the RGB QUAD palette array. The following 2 functions do this by setting up the appropriate data and writing it to the output file.

```
/*=================================================================WRITE_BMINFO==*/
void write_BMInfo(FILE *fp)
{
  int x;
  static char bmi[] = {  40, 0, 0, 0,   /* Bytes in this header  */
                         0, 2, 0, 0,        /* Width in pixels = 512  */
                         0, 1, 0, 0,        /* Height in pixels = 256 */
                         1, 0,              /* Planes = 1           */
                         4, 0,              /* Bits per pixel = 4    */
                         0, 0, 0, 0,        /* Compression = none    */
                         0, 0, 1, 0,        /* Image size in Bytes   */
                         0, 0, 0, 0,        /* Pixels per metre wide  */
                         0, 0, 0, 0,        /* Pixels per meter high  */
```

```
                0, 0, 0, 0,         /* Colours in index used  */
                16, 0, 0, 0         /* Important colours used */
               };

  for (x = 0; x < bmi[0]; x++) {
     putc(bmi[x], fp);
  }
}

/*=============================================================WRITE_RGBQUAD==
*/
void write_RGBQuad(FILE *fp)
{
  int x;
  static char bmq[] = {  0, 0, 0, 0,          /*  0 = Black     */
                0, 0, 191, 0,                 /*  1 = Dk Red    */
                0, 191, 0, 0,                 /*  2 = Dk Green  */
                0, 191, 191, 0,               /*  3 = Dk Yellow */
                191, 0, 0, 0,                 /*  4 = Dk Blue   */
                191, 0, 191, 0,               /*  5 = Dk Magenta*/
                191, 191, 0, 0,               /*  6 = Dk Cyan   */
                192, 192, 192, 0,             /*  7 = Lt Grey   */
                128, 128, 128, 0,             /*  8 = Dk Grey   */
                0, 0, 255, 0,                 /*  9 = Red       */
                0, 255, 0, 0,                 /* 10 = Green     */
                0, 255, 255, 0,               /* 11 = Yellow    */
                255, 0, 0, 0,                 /* 12 = Blue      */
                255, 0, 255, 0,               /* 13 = Magenta   */
                255, 255, 0, 0,               /* 14 = Cyan      */
                255, 255, 255 ,0              /* 15 = White     */
               };

  for (x = 0; x < 64; x++) {
     putc(bmq[x], fp);
  }
}
```

This next function, 'convert_screen' is where the bulk of the hard work is done. A few look-up tables are set up to hold the various values required when converting a QL colour code to a PC colour code. Because we need 4 bits for every pixel in the output file each byte can hold 2 pixels worth of data. This is a lot less than a QL's ability to hold 4 or 8 pixels worth of data, but remember we are dealing with a 16 colour bitmap here.

```
/*=============================================================CONVERT_SCREEN==*
/
void convert_screen(FILE *fp, int mode)
{
  int scan_line,                    /* Loop counter for each scan line     */
     line_offset,                   /* Offset into display memeory for line */
     word,                          /* Loop counter for each word in a line */
     word_offset;                   /* Offset into memeory for each word    */


  unsigned char  red,               /* Odd addresses byte of word offset    */
                 green,             /* Even addressed byte of word offset   */
```

```
top4_nibble[] = {0x00,      /* Black */
                 0x90,      /* Red   */
                 0xA0,      /* Green */
                 0xF0},     /* White */

bot4_nibble[] = {0x00,      /* Black */
                 0x09,      /* Red   */
                 0x0A,      /* Green */
                 0x0F},     /* White */

top8_nibble[] = {0x00,      /* Black  */
                 0xC0,      /* Blue   */
                 0x90,      /* Red    */
                 0xD0,      /* Magenta */
                 0xA0,      /* Green  */
                 0xE0,      /* Cyan   */
                 0xB0,      /* Yellow */
                 0xF0},     /* White  */

bot8_nibble[] = {0x00,      /* Black  */
                 0x0C,      /* Blue   */
                 0x09,      /* Red    */
                 0x0D,      /* Magenta */
                 0x0A,      /* Green  */
                 0x0E,      /* Cyan   */
                 0x0B,      /* Yellow */
                 0x0F},     /* White  */

pixel[8];                   /* One words worth of pixels */
```

Remember that the PC needs its data from the bottom left while the QL's display is arranged from the top left. We need to calculate where the bottom line starts in the QL's memory and convert it first. Then do the next line up and so on. As there are 256 lines in the QL's memory we need to loop around this many times to process every line.

```
for (scan_line = 255; scan_line > -1; scan_line--) {
```

The start position in memory is the screen start plus the scn line times 128 as there are 128 bytes in a scan line. We assume that scan lines are numbered from 0 to 255.

```
line_offset = 131072 + (scan_line * 128);
```

Each scan line holds 128 bytes which is actually 64 words of colour data. It seems appropriate to loop around this many times and process each word. That is what we do now.

```
for (word = 0; word < 64; word++) {
```

We must calculate the QL address for each word. This is simply the scan line address plus the loop index times 2. In C, we can shift an integer to the left to multiply by 2 and this operation is very quick compared with a multiplication.

```
word_offset = line_offset + (word << 1);
```

Now that we have obtained the starting address, we can extract the green and red bytes from it. The next couple of lines look complicated, but are in actual fact equivalent to PEEK.

```
green       = *((unsigned char *)word_offset);
red         = *((unsigned char *)(word_offset +1));
```

Just to show where we are in the conversion, we flip all the bits in the red and green bytes before setting them back into the display. This has the action of inverting the whole screen from the bottom up as we go along.

The following is a quite complicated C expression that is doing a POKE as well as checking for MODE 4 and if not found masking out the flash bits in the green byte of data.

```
*((unsigned char *)word_offset) = mode == 4? ~green: (~green)& 0xAA;
*((unsigned char *)(word_offset +1)) = ~red;
```

The following is an 'aide memoir' for the bit patterns in the QL memory word and what each value means in mode 4 and mode 8.

```
/*========== MODE 4 ==========
* GREEN  ¦ RED ¦ PIXEL COLOUR
*============================
*   0    ¦  0  ¦ 00 = BLACK
*   0    ¦  1  ¦ 01 = RED
*   1    ¦  0  ¦ 10 = GREEN
*   1    ¦  1  ¦ 11 = WHITE
*---------------------------
```

```
*========== MODE 8 ==========*
* GR ¦ RD  BL  ¦ PIXEL COLOUR *
*=============================*
* 0  ¦ 0 ¦ 0   ¦ 000 = BLACK  *
* 0  ¦ 0 ¦ 1   ¦ 001 = BLUE   *
* 0  ¦ 1 ¦ 0   ¦ 010 = RED    *
* 0  ¦ 1 ¦ 1   ¦ 011 = MAGENTA*
* 1  ¦ 0 ¦ 0   ¦ 100 = GREEN  *
* 1  ¦ 0 ¦ 1   ¦ 101 = CYAN   *
* 1  ¦ 1 ¦ 0   ¦ 110 = YELLOW *
* 1  ¦ 1 ¦ 1   ¦ 111 = WHITE  *
*-----------------------------*/
```

Using the table above we can see that if this is a mode 4 picture we need to get each individual bit from the red byte and the corresponding bit in the green byte. We then need to shift the two bits along to the right until they can be OR'd together to form a 2 bit value in the far right of the word. This 2 bit value is simply the colour code 0 to 3 as outlined above.

In mode 4 we can mask out and shift along a set of 8 pixels from each word. We have a LOCAL array called 'pixel' which ends up holding the colour values for 8 QL pixels. The format of each colour value is GR for Green bit and Red bit.

```
if (mode == 4) {

  pixel[0] = ((green & 128) >> 6)    ¦ ((red & 128) >> 7);
  pixel[1] = ((green &  64) >> 5)    ¦ ((red &  64) >> 6);
  pixel[2] = ((green &  32) >> 4)    ¦ ((red &  32) >> 5);
  pixel[3] = ((green &  16) >> 3)    ¦ ((red &  16) >> 4);
  pixel[4] = ((green &   8) >> 2)    ¦ ((red &   8) >> 3);

  pixel[5] = ((green &   4) >> 1)    ¦ ((red &   4) >> 2);
  pixel[6] = ((green &   2))         ¦ ((red &   2) >> 1);
  pixel[7] = ((green &   1) << 1)    ¦  (red & 1);
```

Note how we need to shift the last green pixel in the opposite direction to all the others ? Now that our 'pixel' array holds the QL colour codes for each of the 8 pixels on the screen, we can use our lookup table to convert the QL colour code to a PC palette index, but, each QL pixel maps on to 4 bits of a PC byte so we need to convert a nibble at a time. 8 pixels = 8 nibbles = 4 bytes in the output file.

Pixel[0] is the top half of byte 0, pixel[1] is the bottom half of byte 0 and so on.

```
  fprintf(fp, "%c%c%c%c",
          (top4_nibble[pixel[0]] ¦ bot4_nibble[pixel[1]]),
          (top4_nibble[pixel[2]] ¦ bot4_nibble[pixel[3]]),
```

```
                (top4_nibble[pixel[4]] | bot4_nibble[pixel[5]]),
                (top4_nibble[pixel[6]] | bot4_nibble[pixel[7]]) );

        } else {
```

If this is a mode 8 picture we can see from the above table that we need to get the red and blue bits from the red byte and the corresponding bit in the green byte while ignoring the flash bits in the green byte. We then need to shift the appropriate bits along to the right until they can be OR'd together to form a 2 bit value in the far right of the word. This 3 bit value is simply the colour code 0 to 7 as outlined above.

In mode 8 we can mask out and shift along a set of 4 pixels from each word. This time the LOCAL array called 'pixel' ends up holding the colour values for only 4 QL pixels. The format of each colour value is GRB.

```
pixel[0] = ((green & 128) >> 5)    | ((red & 192) >> 6);
pixel[1] = ((green &  32) >> 3)    | ((red &  48) >> 4);
pixel[2] = ((green &   8) >> 1)    | ((red &  12) >> 2);
pixel[3] = ((green &   2) << 1)    | ((red &   3));
```

Having our lookup tables for the mode 8 pictures helps as we can now convert each of the 4 colour codes for the QL into a palette index for the PC. Once again, 4 bits are used to hold each pixel's index.

Note this time that we are setting 2 pixels worth of data in the output file for each pixel in the input file. This is because we must preserve the QL's display width to height ratio otherwise we would end up with very square looking mode 8 conversions.

To preserve the screen ratio, we simply duplicate each pixel across the screen. (Actually, when this program was first written I did not duplicate the pixel data in the output file. This is why I still use the lookup tables. With a couple of minutes thought you could get rid of the 'top8' and 'bot8' nibble tables and simply use a single table with the duplicated values in it and write out the appropriate value. Try it if you like !)

```
        fprintf(fp, "%c%c%c%c",
                (top8_nibble[pixel[0]] | bot8_nibble[pixel[0]]),
                (top8_nibble[pixel[1]] | bot8_nibble[pixel[1]]),
                (top8_nibble[pixel[2]] | bot8_nibble[pixel[2]]),
                (top8_nibble[pixel[3]] | bot8_nibble[pixel[3]]) );
    } /* end if   */
```

We are now at the end of both of the loops for each word in a scan line and for each scan line in the memory map. Jump back to the appropriate loop and process the next word or scan line accordingly. This is also the end of the function.

```
    } /* end for word  */
  } /* end for scan_line */

}
```

And that is that. If you try to work through the valued in the masks above you will see that we take each red, green (and blue in the mode 8 files) bits out of the memory word, get rid of all the values for other pixels and the make the data into a QL colour code. When we have decodes an entire word, these colour code are converted to a 4 bit PC palette index and are written to the file as a 2 or 4 pairs of nibbles.

You can see that a QL screen dump is very much smaller than a PC bitmap, at least half the size in uncompressed form.

PORTING TO THE PC: Now that you have converted the screen dumps all that remains for you to do is to find some method of converting the file from QL disc format to an MS-DOS disc. I use XOVER from Digital

Precision but you can use DiscOver or Multi DiscOver from Dave Walker or any other QL to IBM converter that you may have. On the QXL I can write the files directly to floppy and then CTRL SCROLL-LOCK back to MS-DOS to copy them onto the hard drive.

If you want to experiment with altering the colours from QL to PC then feel free, just set up a different value for the index number in the lookup tables for each nibble. I actually convert all my QL green pixels into a light grey when I convert my files as this seems to print out better on final copy. Green tends to become a very pale white when converted to mono for the printer.

This is simple to do, for mode 4 set the value in TOP4_nibble[] to 70 hex and in BOT4_nibble[] to 07 hex for the green element in the array. Do the same with the mode 8 values if converting mode 8 screen dumps.

Note that the value is simply the palette index for the bottom nibble arrays and the palette index with a zero on the end for the top nibble arrays. You can have all sorts of fun with different colour schemes if you like.

Finally: I have uploaded via the QBBS bulletin board the source code, a text file and the executable program for Bob to examine. If he thinks that it is good enough I have no objections to the program and source files being included in PD libraries and/or uploaded onto QBOX or whatever. Please feel free to use and abuse this program to your hearts content !

Things in SMSQ/E (Part 2)

Duisburg, GERMANY - Jochen Merz

SER_PAR_PRT - This things controls the serial and parallel ports, and allows access in various ways not defined by trap #2 or trap #3 access calls.

PARU SBASIC equivalent is PAR_USE. Optional call parameter is a string.
PARB - SBASIC equivalent is PAR_BUFF. Optional long word on call.
PARC - SBASIC equivalent is PAR_CLEAR. No parameters (yet, we only have one one PAR port).
PARA SBASIC equivalent is PAR_ABORT. Same as before
PARP - SBASIC equivalent is PAR_PULSE. Requires a word on call.
SERU - SBASIC equivalent is SER_USE. Optional call parameter is string.
SERF - SBASIC equivalent is SER_FLOW. Optional call parameter is long (the port number), followed by character (the flow control).

SERB - SBASIC equivalent is SER_BUFF. Three optional call parameters, all long: port number, send buffer, receive buffer.

SERR - SBASIC equivalent is SER_ROOM. Two optional parameters on call, both long: port number and room. SERE - SBASIC equivalent is SER_CDEOF. Two optional paramaeters on call both words: port number & time.
SERC - SBASIC equivalent is SER_CLEAR. Optional word defines port.
SERA - SBASIC equivalent is SER_ABORT. Same as above.
PRTU - SBASIC equivalent is PRT_USE. Optional call parameter is string.
***PRT$*** - SBASIC equivalent is PRT_USE$. It returns a string.
PRTB - SBASIC equivalent is PRT_BUFF. The optional call parameter is long, the buffer.
PRTC - SBASIC equivalent is PRT_CLEAR. No parameters.
PRTA - SBASIC equivalent is PRT_ABORT. No parameters.
SERP - SBASIC equivalent is SER_PAUSE. Two optional words on call, port and pause.
KBD - deals with keyboard settings. At the moment, all you can do is set the keyboard table:
KBDT - two unsigned long call parameters. The longs are pointers to the keyboard table and the non-spacing ident table. Unfortunately, the definition is quite complex so that it cannot be published here.

QVME - the QVME things had better been called display control, because that's what it is. As the QVME card was the first flexible display adaptor which was used for QDOS/SMSQ, it was named after it. Now, the QVME-Thing provides access to display control over the QVME card, the Extended4-Emulator, the monochrome mode and the QXL. You can access any extension in the QVME-thing, even if the hardware does not exist to support it. Inverse, for example, is supported only in monochrome mode on the ATARI - if you call DISP_INVERSE on other machines, it is simply ignored. It just tries to do its best, without guarantee.

INVR - this is DISP_INVERSE. Will invert monochrome display mode on the ATARI. Optional word defines state of inverse.

TYPE - returns the display type in a signed word. Currently, you can get the following values: 0 for old QL-emultor, QXL and, of course, QL. They all handle standard QL sizes in MODE 4 and 8. 1 Extended , 4-QL Emulator, 2 QVME and 4 ATARI Monochrome mode.

SIZE - sets the display size if possible. It accepts up to 6 parameters, all optional words. For the detailed meaning of the parameters, refer to the SMSQ/E manual.

RATE - sets the display rate. It accepts up to 4 optional word parameters.
BLNK - sets the horizontal and vertical blank. Two optional word parameters are accepted.

HOTKEY - is the Thing which allows access to the HOTKEY System II. It is not different to the separate HOTKEY System II which can be used on non-SMSQ/E systems as well, and the complete documentation is far too much to fit in here. However, if you wish to access the HOTKEY System II in machine code, you will find all the internal vectors and parameters explained in the QDOS/SMS Reference manual.

NEWS Italian Style !!

Reggio Emilia, ITALY - Davide Santachiara

[] The VII Italian QL meeting: The 7th Italian QL meeting was held in Reggio Emilia Italy. The meeting was attended by about 40 people. This is really a good result for various reasons. First of all the long awaited new hardware pieces were not available for the meeting - think about SuperHermes, the QL Graphic Card or the QXL Gold. In second instance we heared only in the last days before the meeting of the attendance of Jochen Merz with all his huge software catalogue. So people were not informed about this news. Finally the period of the year is not the best because of possible fog and cold temperature. Anyway the weather on Sunday luckily was sunny.

Anyway the meeting was a good success and we heared a lot of news about forthcoming hardware releases from Stuart Honeyball (alias Miracle systems) and Zeljko Nastasic (alias Nasta). I expect that most of these products will be available for the next Italian meeting (that should be held around May/June) and I am quite sure that the 8th Italian meeting could be named "QL: the revenge".

Have you ever been to Italy ? Maybe the next meeting could be the occasion to visit this wonderful country. In June the weather is warm, the food is always fantastic and due to the geographic position of Reggio Emilia it is quite easy to reach beatiful towns such as Bologna, Milan, Florence or Venice by train. If you prefer the seaside (Mare Adriatico o Ligure), the lake (Garda lake) or the mountains: all are reachable in less than 2/3 hours.

Who attended the meeting ? Stuart Honeyball (and his father) of Miracle Systems from the UK, Zeljko Nastasic from Croatia representing also Ron Dunnett's Qubbesoft PD, Jochen Merz came from Duisburg Germany with all his software, Bill Richardson from the UK (via Rome where he stayed before the meeting for some days), QItaly club (the Italian QL club) represented by "the president" Roberto Orlandi, QItaly magazine director Dr.Eros Forenzi, the hardware problem solver Romaldo Parodi and "il tuttofare" Roberto Calcagno ("tuttofare" is an Italian expression to indicate somebody who does all sorts of works), Daniele Terdina, author of Q-Emulator, QL emulator for the MAC; Ergon ie myself (Davide Santachiara) and Marco Ternelli from Reggio Emilia.

[] Miracle Systems - Stuart showed a prototype of the QXL GOLD (not working yet). This interface will allow to connect Gold Cord or Super Gold Card on PC ISA bus. In practice you will get a sort of QXL with reduced speed. With this card you will be able to use the PC peripherals such as mouse, keyboard, hard disk, CD rom, graphic card from your QL environment. The operating system supplied with the QXL GOLD will be another version of SMSQ. Perhaps a specific SMSQ/E version for the QXL GOLD will be released by Jochen Merz.

With a Gold Card you will be able to use QL or EGA resolution. With the Super Gold Card you will be able to use QL, EGA, VGA or SVGA resolutions such as on the QXL.

The QXL GOLD will use "bus mastering" to communicate with the PC, so I/O speed should be higher than on the QXL. Stuart said that screen refresh should be smoother than on the QXL, and probably also (hard) disk, serial and parallel i/o should be faster. The QXL GOLD will not have network ports. The QXL GOLD should be available in the near future.

Miracle is also planning a PCI QXL with 68060 by late '96, with SIMM slot for ram. Should be 3-4 times faster than the standard QXL, but not so cheap. It will come with no memory because it will use the PC's own memory. Anyway for higher performance you will be able to add your own 72pin SIMM on the PCI QXL board. I think that at the moment this project is simply an idea.

[] Zeljko Nastasic / Qubbesoft P/D - Nasta as usual had a lot of interesting news, project both in his mind or already "alive & kicking".

* the graphics card is to be released by QUBBESoft/Nastasic, and will be much better that the original MasterPiece. Resolutions up to 1024x768 with some in between (eg. 640x480), and also 16 and 256 colours! It will support many kinds of monitors, and at high vertical frequencies. You'll need a Super GC for higher graphics, but even with just a Gold Card you'll be able to use analog monitors (better video quality and reliability over old QL monitors). The graphics card replaces the QL motherboard and will not support Microdrives and TV output.

* a successor to the Super Gold Card is being considered. Three times as fast and with SIMM slot for memory. A new powerful RISC processor from Motorola (ColdFire) will be used. This has roughly the same performance of a 68EC040 but at a much lower price.

* Syquest removable drives work perfectly with QUBIDE. Every disks can hold more than 200 Mb of data. This may be used both for backups or as a true hard disk (it is not much slower than a standard HD). CDROM support for QUBIDE is only a matter of writing a proper driver. This is now in the hands of Phil Borman, maybe this will happen in the near future ? (The driver is ready NOW!)

* Zeljko had on show a graphic card to drive an LCD monitor from the QL. It was perfectly working in VGA resolution. This graphic card currently will not be marketed because the other projects (new QL motherboard/graphic card) have currently higher priority. Anyway it is possible that in the future this project will be commercially available. Maybe with the reduced dimensions of the new QL motherboard it would be possible to have a real portable QL.

* Obviously the other first class products designed by Zeljko and sold by Qubbesoft were available: The Qubide QL AT/IDE interface, QPLane, powered back plane for the QL, and the cable to connect more than one HD to the Qubide without having to worry about master/slave HD settings.

[] Daniele Terdina with his Q-Emulator for the MAC - This program allows to emulate a QL on the MAC. It needs MacOS 7.0 or higher, 4 Mb ram, colour monitor, Mac with 68030 or 68040 processor. Q-Emulator emulates the 68008 processor, QL video, keyboard, sound, serial ports, file system 1 (no support for subdirectories currently) and can read QL disks. Mode 8 does not support flash. The emulated QL can have 128Kb to 4 Mb memory. It is also possible to plug ("virtually") 16Kb eprom expansions.

QL compatibility is almost >99%. Only 3 games do not run because of their bugs which are not critical on the QL but are difficult to be trapped on the emulator. Just to say how good is the compatibility of this product just think that Daniele is using his QEmulator with the extended environment, the QBOX package and QPoint as a "point" of Ergon BBS. In fact you can contact Daniele also in the International QL Fidonet message area.

The QL Rom image is not included in the emulator for copyright reason and it must be copied on the MAC before running Q-emulator. On a QL it can be saved to disk with a command such as SBYTES flp1_ram,0,49152 provided you have not a [Super] Gold Card fitted which does some patch in the ROM area. AH, JM, JS, MGx roms are fine. Maybe also Minerva will be supported in the future. Speed is that of a standard 68008 QL on a MAC with 68040 at 33 MHz.

Q-Emulator is a commercial program and costs 50.000 ITL (Italian Lire) (30.000 ITL for unemployed students) plus post & packaging. This means slightly more than $30 or 20 pounds. You will pay when receiving the goods in your currency. The commercial package includes a 14 page manual, instructions for installation, a disk with the full working version plus a demonstrative "lite" version, some freeware QL programs, general informations about the QL and part of the C source code (the 68008 emulator).

Minor updates (1.x) will be available freely on Internet or on BBS. V2.0 versions (such as the version for Power MAC) will be seperate products but already registered users will have huge discounts.

A freely distributable version (Q-Emulator Lite) is already available on Internet and on some BBS. This includes a freeware version with less options and 1/4 the speed of the commercial version. This requires MacOS v6.0.7 or higher, at least 2 Mb of ram and any type of monitor (so it has less requirements than the commercial version).

Projected developments: As soon as Daniele gets the PowerPC C compiler a PowerMAC version of Q-Emulator will be released. He is also thinking about a re-write in 68000 machine code of the C version to get higher speed (and perhaps also a PowerPC m/c version). To order Q-emulator or to get more information write to:

Daniele Terdina, Via dei Navali 16/1, 34143 TRIESTE, ITALY.
Internet E-mail: sistest@ictp.trieste.it
Fidonet address (Netmail) : 2:332/811.5

[] Jochen Merz - Jochen was the special guest. He arrived with his Audi and an Italian interpreter too! Well, really he was a friend of Jochen who was born in Italy but who has been living in German for years. He had SMSQ/E v2.70 and all the other stuff. The new program he had in the catalogue were version 8 of QD, LDUMP and IO/2. LDUMP is a program to make exact screen dumps on dot matrix, inkjet or laser printers. IO/2 is a collection of over 350 new SuperBasic keywords. It is interesting that the package includes the assembly sources. Sorry Jochen, I did not buy them, but I bought WinED and I was able directly at the meeting to recover some files of a faulty hard disk (QL+Qubide system) in conjunction with WinEditor by Phil Borman (which is given with the Qubide interface). WinED is incredibly fast in finding strings on the HD while Phil Borman's WinEditor has a quite powerful "collect" function to recover files.

QL software Emulator for PC - At the Munich meeting in September 1995 was shown a QL emulator running on a 486 PC. This was quite interesting because it was software only! Jochen Merz has obtained the rights and probably the package will be sold as a commercial program and the operating system obviously will be SMSQ/E.

[] Bill Richardson - Bill had some Z88 notebook for sale. The Z88 computer has a built-in wordprocessor, spreadsheet, databse and organizer. It is possible with a PD software to transfer software to/from the QL via the serial port. If you need a portable computer consider the Z88 because it is very cheap.

[] Myself - Finally some notes about myself. In the last months I had very few time for programming for various reasons: mainly this was due because of my last engineering examinations and my work on the thesis. I am doing it at the Space Division of Ferrari, the same Ferrari known as Formula 1 team and I will take the degree on the 20th of March. Another reason is that last year I set-up a QL based Fidonet BBS (Ergon BBS, whose first birthday was on 25th of December) and this requires quite a lot of time to be maintained (just think at all the new PD stuff which is available weekly and that I have to check and put in the right file area with a detailed description). I have also some responsabilities as Fidonet node and so I must be very careful in all my operations. Here in Italy we were able to convince Fidonet local "chiefs" to open an Italian message area (something like an internet newsgroup) exclusively dedicated to Sinclair or compatible computers. This has allowed to find some lost QL Italian users which did not know that from the death of the Sinclair the QL has continued to live and that many new hardware and software products are available.

Ergon BBS is currently based on Jan Bredenbeek's QBOX but I am now switching to the very powerful PBOX from Phil Borman. The BBS is running on a SuperGold Card QL with Minerva rom (or SMSQ/E v2.70), 210 Mb Seagate Hard Disk, Qubide interface (rom v1.37) and SuperHermes, all mounted into a System 2 SPEM cabinet. The modem is a ZyXEL Elite 2864 (v34). I can recommend all the above stuff. SuperHermes fast serial port prooved to be very reliable and fast. I also connected a mouse to SuperHermes and it works perfectly. I am not using a PC keyboard because I am still using the SPEM futura keyboard which is more than adequate for me.

Thanks to SuperHermes fast serial port I am finally able to use the alfa version of QVM from Jonathan Hudson. This program (used in conjunction with a ZyXEL modem and QFAX) allows to use your QL as an answering machine or fax machine (and even data calls can be handled correctly). Currently there are some minor problem that surely Jonathan is able to fix but the program essentially works. In case the call is a fax the modem is able to distinguish the CNG fax tone, a specific message is sent to QVM and QFAX is invoked. Instead if the call comes from a human, a message can be recorded to the QL Hard Disk and later listened. QVM is powerful, also because you can remotely hear recorded message (ie calling your phone number from outside and using an appropriate tone composition). If speed is not all for you, you can find second hand ZyXEL v32bis models (U-1496 - upto 14400 or 19200 with other ZyXEL) at very reasonable price in the USA. This modem is incredibly robust (I used it for many years) and works perfectly with the QL and in particular with QVM and QFAX (Jonathan Hudson has got that modem too).

You can give a call to Ergon BBS every day from 21:00 to 03:30 CET at the following phone number: +39 522 300509 (fax calls are allowed too). Instead my voice/fax/answering machine number is +39 522 300409. The fax/answering machine is available from 10:00 to 19:00 CET. I have also a new Internet account. You can contact me at the following E-mail address: ergon@xmail.ittc.it

I forgot to say I have also bought a PC (it was absolutely necessary for my thesis and for Internet connection) but obviously I have also bought a 4Mb QXL. As I already had a full blown 68040 (ie. a 68040 with memory

management unit and floating point unit) I bought the QXL without processor. I mounted my CPU, fitted the QL in the ISA slot, executed SMSQ from a DOS session and voila', the QL screen appeared on the PC monitor. The QXL card is plug & play even without Windows 95. As my CPU can run at 33 MHz I have bought a proper oscillator and fitted it on the QXL. I have also put a small fan on the CPU. The QXL works perfectly at 33 MHz, and as you can imagine, it runs 65% faster than a standard QXL which runs at 20 MHz. The only minor problem is that the network port does not work anymore. Anyway when I need it (mainly for backup purposes of my SGC QL) I simply change the oscillator. There is also an added bonus: the latest version of C68 (v4.20b) fully supports CPU with floating point unit. So all programs which make an heavy use of floating point calculation, when recompiled with that C68 version, will (hopefully) run even faster.

As my PC is fitted with 16 Mb of ram (which currently seems the minimum to get acceptable results with Windows 95) the QXL DOS batch file loads a DOS ramdrive and format it at 4 Mb. In this way in the QL environment I have 4Mb of own ram plus 4 MB of external ramdisk which is very useful to store temporary data. The external ramdisk is seen on the QL as win3_, you should format it in your boot file with a command such as format win3_4 before using it!

To increase the speed in disk handling I use the very useful HyperDisk DOS utility (with roughly 4 Mb of floppy/hard disk cache). In this way I obtain something which is quite similar to background floppy/hard disk operations. Apart this disk operations are speeded-up a lot both when using QL or DOS disks. It is a pity that i/o operations are so slow on the QXL. HD speed is acceptable, disk handling is OK provided you use HyperDisk, but the speed of the parallel port (2 Kb/s againg >40 Kb/s of the SGC parallel port) and that of the serial port (maximum 2 Kb/s) is quite limiting for some of my purposes. This should be improved with the QXL GOLD because it will use bus-mastering.

Finally the batch file loads a DOS utility which switch off the monitor (my monitor has a power saving function) in case no key is pressed (or mouse is moved) for 5 minute. This utility works perfectly in QL mode and so I do not need any QL screen saver.

That's all for know. I wish all the best to all IQLR readers and to all the QL friends I meet at the various meetings in Italy or in Europe.

Notes & News From JMS

Duisburg, GERMANY - Jochen Merz

JMS-BOX 1 & 2 : It has happened! The old QDOS/SMS-Box (phone-number 502013) has been renamed to JMS-BOX 1 and it is still running under the old QBOX program. It is still maintained, but most maintainance effort will be put into the new box. JMS-BOX 1 is now running on a TT, with much more harddisk space and much faster! There is still a ZyXel modem connected, so you can get into it at 14400 with any modem or 16800 with other ZyXels. JMS-BOX 1 will be converted when the new box 2 has proved that it runs 100% stable.

The JMS-BOX 2 (phone number 502014) is also online now, it is running under Phil Borman's PBOX system (with many advantages), and it is running on the same machine. This will allow data sharing between the two boxes in the future, when both boxes run PBOX. At the moment, the highest connect you can get at this box is 19200, but this is still 25% faster than JMS-BOX 1. It will change to 28800 soon, hopefully beginning of March, as we seem to have a small problem with higher BAUD-rates at this port.

Boston Show: Everybody from Germany, The Netherlands, Belgium, Switzerland, Austria etc. is welcomed to contact me for a joined travel to Boston, I'm happy to organise it from here. The price from Amsterdam would be 799,- DM plus US taxes. Whoever is interested should contact me as soon as possible.

Updates and News: First of all, SMSQ/E is Version 2.72. Major new features are event handling between jobs, even between SBASICs. The events can be handled on job-level and even on window-level. If you wish to get a free update, just send disk and return postage (IRC's) or download it from the box. If you wish to use the events for your own programs, then I recommend you also order a new manual (DM 16,-).

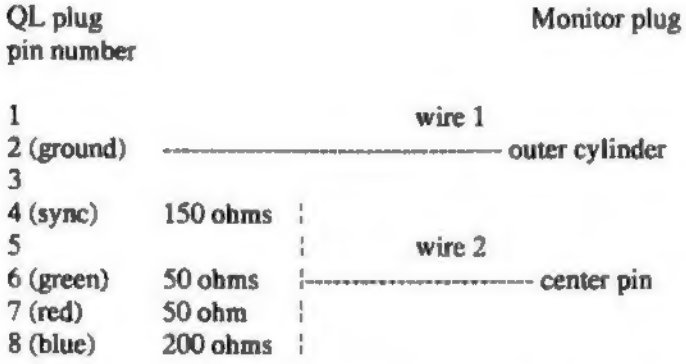
QPTR is upgraded to handle the events on the pointer-level, so if you wish to use events in your pointer-programs then you should either get an update of QPTR (DM 16,-) which explains the machine-code aspects of event handling, or you will get the documentation automatically with the next, soon coming update to the QDOS/SMS Reference manual (provided you booked it).

Menu and QMenu is V6.27 and it will now handle any device or directory in the directory-select menu. In the past, it was only used for directory-select and therefore always added an underscore at the end of the filename, but this wasn't very much use when you typed in SER or PAR, 'cause it always returned SER_ or PAR_. This is changed. Directory-select now recognised directory devices and adds the underscore just in case it is one. So you can easily enter SER, PAR, n1_history_fred etc. to be the destination device.

Monochrome Monitors (Making YOURS Brighter)

Munich, GERMANY - Franz Krojer

I have a PC with QXL-Card and SMSQ/E. But I still use a normal German QL with floppies, Trumpcard and a monochrome monitor. It makes no noise and is therefore good for thinking. But with my monochrome monitor (Philips) I had the problem, that the screen was not bright enough for many of my programms. I couldn't see parts of the text, red lines or even the cursor. A comfortable use of Quill wasn't possible, for example. And I know, that other people have similar problems with their monochrome monitor. But there is a simple solution. All what You need - additionally to Your QL monitor cable - are 4 resistors (50-200 ohms, 1/4 watt) . After that, the range of contrast and brightness of YOUR monochrome monitor will be significantly enhanced.



You simply go with wire 1 directly from the ground pin of the QL plug to the outer cylinder of the monitor plug. (If You use the original QL monitor cable, the pin 2 (ground) is yet connected with the outer cylinder.) Then You link the pins 4, 6, 7 and 8 of the QL plug with the 4 resistors. These 4 resistors You link with wire 2, which then is connected with the center pin of the monitor plug (when using the original cable it is yet connected). Of course, You can modify the values of the resistors a little. Perhaps You get, for example, better results if You take 100 or 300 ohms for pin 8 (blue) of the QL plug. One notice: I am not sure, if the pins of the QL plug are the same for German, English, French or American QL's. But Your QL manual should mention what pins are correlated with "ground", "sync", "red", "green" and "blue".

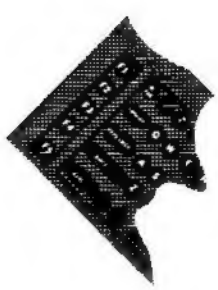
About the Cover

After the recent Hove (UK) show the group in the photo ended up in Roy Wood's kitchen. The computers in the Photo are an Atari TT, Epson PC and an Atari Stacy and just out of camara range was a Tower Cased QL. ALL four systems were running SMSQ/E.

From Right to left are: Jochen Merz , Joachim van der Auwera, Tim Wood, Roy Wood (the host) and Steve Wollington.

Q Branch

Feeling out on a limb ?
Reach out for Q Branch
Suppliers of Quality QDOS / SMSQware



P.O. Box 7
Portslade
East Sussex
BN41 2ND
ENGLAND

Tel. 01273-386030
Fax 01273-381577
U.K. code (44)

PRICES:

QD	£ 58.00	QPAC 1	£ 20.00
QBASIC	£ 23.00	QPAC 2	£ 40.00
LIBERATOR	£ 50.00	QTYP 2	£ 30.00
QD + QBASIC	£ 75.00	QLOAD/QREF	£ 15.00
QD + QBASIC + QLIBERATOR	£ 120.00		
BUDGET QLIB	£ 25.00	DISA 2	£ 40.00
EASYPTR 3 pt 1	£ 40.00	CUESHELL 2	£ 40.00
EASYPTR 3 pt 2	£ 20.00	DISK MATE 5	£ 37.00
EASYPTR 3 pt 3	£ 20.00	LDUMP	£ 30.00
FiFi	£ 24.00	I/O TOOLKIT	£ 46.00
QPTR	£ 42.00	QDOS/SMS ref. manual	£ 38.00
QSUP	£ 36.00	QLERK	£ 25.00
QLQ	£ 32.00		
EPROM MANAGER	£ 28.00		
LINEDESIGN	£ 100.00		
QMAKE	£ 21.00		
QMENU	£ 20.00	DATADESIGN	£ 67.00
WINED	£ 24.00	pfDATA	£ 24.00
QSPREAD	£ 77.00	pfLIST	£ 24.00
QMON/JMON	£ 44.00	PROforma	£ 112.00
DATADESIGN API	£ 24.00	FONTPACK	£ 89.00
pfb2pfl	£ 66.00		

ARCHANOIDS	£ 17.00	DIAMONDS	£ 17.00
FIREBIRDS	£ 17.00	BOB'S PIPES	£ 20.00
SUPER GAMES PACK	£ 40.00	QSHANG	£ 22.00
THE LONELY JOKER 2	£ 26.00	MINEFIELD	£ 19.00
BLACK KNIGHT	£ 55.00	DOUBLEBLOCK	£ 20.00
THE ORACLE	£ 24.00		

The Wood and Wind Computing Archive Suite :-

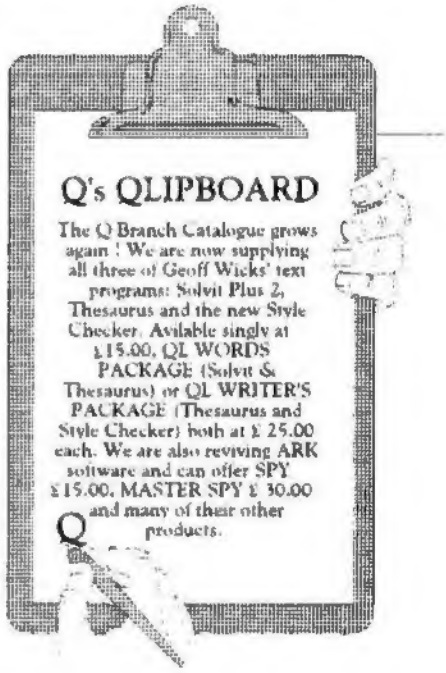
DB Progs	£14.00		
DB Easy	£18.00		
QLERK	£ 25.00	Printed Manual for QLerk	£ 20.00

Albin Hessler's SERmous.
Software and Black mouse £ 40.00

SMSQ/E

Gold Card / Super Gold Card version	£ 90.00
Atari version(if you already have the QJUMP drivers C, D, or E)	£ 90.00
Atari version (incl drivers)	£ 112.00
Atari Monochrome screen version	£ 134.00
QXL version	£ 90.00
All three versions at a budget price	£ 180.00

We can accept cheques in Sterling either drawn on a U.K. bank or Eurocheques. European orders under £ 50.00 add £ 3.00 post and packing. Orders over £ 50.00 are post free. Orders outside the EEC under £ 100.00 please phone for postal rates. Other orders are post free.



Q's QLIPBOARD

The Q Branch Catalogue grows again ! We are now supplying all three of Geoff Wicks' text programs: Solvit Plus 2, Thesaurus and the new Style Checker. Avilable singly at £15.00, QL WORDS PACKAGE (Solvit & Thesaurus) or QL WRITER'S PACKAGE (Thesaurus and Style Checker) both at £ 25.00 each. We are also reviving ARK software and can offer SPY £15.00, MASTER SPY £ 30.00 and many of their other products.

DON'T FORGET ! Our catalogue is not just a list of products and prices. We try to describe each program so that you can choose the items that you need. Why not send us a stamped, addressed, A5 envelope and see for yourself ? This advert was produced using LINEDesign.